НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Комп'ютеризовані системи управління»
спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»
на тему: «Автоматизована система контролю операцій вендингових точок»

Виконав (-ла):
студент (-ка) IV курсу, групи ІА-з61
Балашов Олексій Валерійович _____

Керівник:
асистент кафедри АУТС
Майер Ілля Сергійович _____

Рецензент:
фахівець з тестування програмного забезпечення,
канд. фіз.-мат. наук, АТ “ПУМБ”
Максименко Ігор Іванович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 151 «Автоматизація та комп'ютерно-інтегровані технології»
Освітньо-професійна програма «Комп'ютеризовані системи управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

ЗАВДАННЯ
на дипломний проєкт студенту
Балашову Олексію Валерійовичу

1. Тема проєкту «Автоматизована система контролю операцій вендингових точок», керівник проєкту Майер Ілля Сергійович затверджені наказом по університету від «12» травня 2020 р. №1089-с

2. Термін подання студентом проєкту «09» червня 2020 р.

3. Вихідні дані до проєкту: операційна система – Windows XP/7/8, Linux, MacOS; середовище програмування Intellij WebStorm; мова програмування – TypeScript.

4. Зміст пояснювальної записки: аналіз вимог до автоматизованої системи: основні визначення та терміни, опис предметної області, розробка функціональних та нефункціональних вимог; моделювання автоматизованої системи: моделювання та аналіз програмного забезпечення; опис програмного

та технічного забезпечення: аналіз відомих технічних та хмарних рішень, засоби розробки, технічні рішення, архітектура; аналіз якості та тестування програмного забезпечення; розгортання та впровадження автоматизованої системи.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо): діаграма розгортання; діаграма бази даних; діаграма класів; діаграма варіантів використання; структурна схема

6. Дата видачі завдання «06» березня 2020 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення предметної області	до 13.03.2020	
2	Аналіз існуючих методів розв'язання задачі	до 20.03.2020	
3	Постановка та формалізація задачі	до 27.03.2020	
4	Аналіз вимог до автоматизованої системи	до 10.04.2020	
5	Моделювання системи	до 17.04.2020	
6	Обгрунтування використовуваних технічних засобів	до 24.04.2020	
7	Розробка архітектури системи	до 01.05.2020	
8	Розробка програмного реалізації системи	до 08.05.2020	
9	Налагодження системи	до 15.05.2020	
10	Виконання графічних документів	до 25.05.2020	
11	Оформлення пояснювальної записки	до 25.05.2020	
12	Подання ДП на попередній захист	до 25.05.2020	
13	Подання ДП рецензенту	10.06.2020	
14	Подання ДП на основний захист	15.06.2020	

Студент Олексій БАЛАШОВ

Керівник Ілля МАЙЕР

АНОТАЦІЯ

Пояснювальна записка дипломного проекту містить 144 сторінки, 5 розділів, 20 рисунків, 27 таблиць, 28 посилань.

Об’єкт дослідження: веб-додаток, що можуть використовуватися для контролю та аналізу операцій вендингових точок на підприємстві.

Мета дипломного проекту: полегшити аналітику та оптимізувати логістику товарів до венгдингу.

У першому розділі виконано аналіз предметної галузі, відомих технічних рішень, сформульовано функціональні та нефункціональні вимоги до розробленого програмного забезпечення.

У другому розділі описано бізнес-процеси застосунку, моделювання.

У третьому розділі описано засоби розробки, архітектуру, та інструкцію користувача для впровадження веб-додатку. Було використані хмарні технології на базі платформи Amazon Web Services (AWS).

У четверному розділі описано аналіз якості та специфіку тестування програмного забезпечення.

У п’ятому розділі описано методи впровадження та супроводу програмного забезпечення.

У додатках наведено: текст програмних модулів, схему моделі GraphQL, схема структурна функціональних вимог, схема структурна бізнес-процесів, схема структурна класів програмного забезпечення, схему бази даних.

Ключові слова: angular, веб-додаток, aws, вендинг, програмне забеспечення, аналітика.

ABSTRACT

The explanatory note of the diploma project contains 144 pages, 5 sections, 20 figures, 27 tables, 28 references.

Object of research: a web application that can be used to monitor and analyze the operations of vending points in the enterprise.

The purpose of the diploma project: to facilitate analytics and optimize the logistics of goods for wending.

The first section analyzes the subject area, known technical solutions, formulates functional and non-functional requirements for the developed software.

The second section describes the business processes of application, modeling.

The third section describes the development tools, architecture, and user instructions for implementing the web application. Cloud technology based on the Amazon Web Services (AWS) platform was used.

The fourth section describes the analysis of the quality and specifics of software testing.

The fifth section describes the methods of software implementation and maintenance.

The appendices contain: the text of software modules, the scheme of the GraphQL model, the scheme of structural functional requirements, the scheme of structural business processes, the scheme of structural classes of software, the scheme of the database.

Keywords: angular, web application, aws, vending, software, analytics.

Номер рядка	Формат	Позначення	Найменування	Кільк. аркушів	Номер екзем.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІА361.050БАК.005 ПЗ	Пояснювальна записка	14		
6	А3	ІА361.050БАК.005 Д1	Автоматизована система	1		
7			контролю операцій			
8			точок. Діаграма варіантів			
9			використання			
10	А3	ІА361.050БАК.005 Д2	Автоматизована система	1		
11			контролю операцій			
12			точок. Діаграма розгортання			
13			системи			
14	А3	ІА361.050БАК.005 Д3	Автоматизована система	1		
15			контролю операцій			
16			точок. Діаграма моделі бази			
17	А3	ІА361.050БАК.005 Д4	Автоматизована система	1		
18			контролю операцій			
19			точок. Діаграма класів			
20	А3	ІА361.050БАК.005 Д5	Автоматизована система	1		
21			контролю операцій			
22			точок. Структурна схема			
23						
24						
25						
26						
27						
28						

Зм.	Аркуш	№ докум.	Підпис	Дата	ІА361.050БАК.005 ТП		
Розроб.		Балашов О.В.			Автоматизована система контролю операцій вендингових точок. Відомість технічного	Літ.	Аркуш 1 / Аркушів
Перевір.		Майер І.С.				Т	
Н. Контр.						КПІ ім. Ігоря Сікорського ФІОТ, гр. ІА-з61	
Затв.							

Пояснювальна записка
до дипломного проєкту
на тему: «Автоматизована система контролю операцій вендингових точок»

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

UI	Інтерфейс користувача
UX	Досвід користувача взаємодії з інтерфейсом
CSS	Каскадна таблиця стилів
SCSS	Метамова на основі CSS, призначений для збільшення рівня абстракції CSS коду та спрощення файлів каскадних таблиць стилів
HTML	Стандартизований мову розмітки документів
HTTP	Протокол прикладного рівня передачі даних у мережі
REST	Архітектурний стиль взаємодії компонентів розподіленого додатка в мережі
GRAPHQL	Мова запитів і маніпулювання даними з відкритим вихідним кодом для API, а також Виконавча для виконання запитів з існуючими даними
API	Набір програмних інтерфейсів для взаємодії різнотипного програмного забезпечення, веб-сервісів
SPA	Одно сторінковий додаток (англ. single-page application) - це веб-додаток або веб-сайт, який взаємодіє з веб-браузером, динамічно переписуючи поточну веб-сторінку з новими даними з веб-сервера, замість методу браузера за замовчуванням завантаження цілих нових сторінок
MPA	Багатосторінковий веб-додаток або веб-сайт
MVC	Патерн модель-сторінка-контролер
DOM	Об’єкта модель документу сторінки браузера
CLI	Командний інтерфейс користувача и комп’ютера

NPM	Менеджер пакетів для мови програмування JavaScript, який входить у склад Node.JS
NODE.JS	Програмна платформа створена для перетворювання JavaScript з вузькоспеціалізованої мови у мову загального призначення, реалізована на движку V8
JSON	Текстовий формат обміну даними, заснований на JavaScript
RxJS	Бібліотека для роботи с асинхронними потоками даних
Framework	Програмне забезпечення, що спрощує розробку складних систем
SDK	Набір засобів розробки, який дозволяє розробникам створювати додатки
NoSQL	База даних, спеціально створена для зберігання, створення та отримання певних моделей даних за допомогою гнучких схем
Web-додаток	Прикладна програма, яка зберігається на віддаленому сервері та передається через Інтернет за допомогою браузера
Open Source	Програмне забезпечення з відкритим кодом
ШІ	Штучний інтелект

ВСТУП

В даний час все більше великих або малих підприємств мають змогу створити на налаштувати автоматизовані системи управління, що значно полегшує управління бізнес процесами та швидким збереженням, обробкою та аналізом даних підприємства, таких як логістика, управління складом товарів тощо. На основі приведених даних підприємство може легко на швидко виявляти недоліки тих або інших факторів підприємства для оптимізації та максимальної продуктивності. Автоматизована система управління зберігає всю інформацію в хмарному сховищі, що в свою чергу, допомагає полегшити управління даними на підприємстві.

Актуальність: часто складні та громіздкі готові рішення, являються складними до налаштування, а іноді потрібно інтегрування інших автоматизованих систем, для рішення певної задачі. Що призводить до не оптимізованої автоматизованої системи, яка може призводити до сповільнення роботи даної системи. З метою подолання даної проблеми було створено веб-додаток для клієнтської взаємодії та серверну частину, яка знаходиться у хмарі та має гнучку можливість масштабування системи.

Вендинговий автомат - це автоматизована машина, яка надає споживачам товари, такі як закуски, напої, цигарки та лотерейні квитки після того, як в автоматі сплачено кошти або розраховано кредитною чи дебетовою карткою або спеціально розробленою карткою, тощо [1].

Мета даного проєкту – створення автоматизованої системи контролю операцій вендингових точок (один або група вендингових автоматів), що допоможе в управлінні логістикою по поставці товарів до вендингу і вчасного обслуговування та автоматизації інших процесів.

Дипломний проєкт складається з наступних розділів: вступ, аналіз предметної області, моделюванням програмного забезпечення, опис програмного та технічного забезпечення, розробка програмного забезпечення, аналіз якості та тестуванням програмного забезпечення, висновки, перелік

посилань із електронних джерел, додатків програмного коду та схем моделі бази даних. Графічна частина включає: структурні схеми, діаграму бази даних, діаграми класів, діаграми архітектури системи, діаграми варіантів використання.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОРІВНЯЛЬНА ХАРАКТИРИСТИКА ІСНУЮЧИХ РІШЕНЬ

1.1 Аналіз та порівняння існуючих рішень

У наш час існує велика кількість сервісів та веб-додатків які дозволяюсь вам вести онлайн контроль операцій, товарів, та безліч інших функцій. Все це показує результат, але деякі системи можуть буди складними для інтеграції у вендинг. В даному розділі будуть проаналізовані технічні рішення даної системи. А також будуть виділені їх основні переваги та недоліки.

1.1.1 Аналіз відомих програмних продуктів

В наш час існує досить багато сервісів, систем та платформ для автоматизації процесів підприємства. Розглянемо найбільш популярні з них.

1С Підприємство. Платформа поділена на модулі, які є технологічними прикладними рішеннями, що реалізує один з багатьох загальних функцій. Загальна функціональність системи залежить від конфігурацій одного або багатьох прикладних рішень. Сама платформа не є програмним продуктом для використання кінцевими користувачами, які зазвичай працюють з одним з багатьох прикладних рішень (конфігурацій), розроблених на даній платформі. Конфігурація - поділяється на два види. Перший вид конфігурації дозволяє встановити статичні змінні що характеризують підприємство та сферу діяльність. Другий дозволяє конфігурувати процеси у вигляді програм на вбудованій мові високого рівня, який характеризує бізнес-процеси в динаміці. Таким чином, це дозволяє автоматизувати різні сфери діяльності, за допомогою єдиної технологічної платформи. Система програм "1С:Підприємство" допомагає автоматизувати діяльність організацій та приватним особам. До недоліків системи можна віднести високу вартість впровадження та експлуатації, а також надто складну будову конфігурацій, для освоєння якої

потребується тривале навчання спеціалістів. Для автоматизації бізнес-системи і виконання масштабних та складних систем, рекомендується звертатися до партнерсько-дистрибуторської мережі. У деяких прикладних рішеннях можуть буди відсутня система захисту інформації від несанкціонованого доступу, що призводить до неможливості використовувати дану систему де потрібно зберігати досить дані [9, 10].

SAP ERP. Система програмного забезпечення, завдяки якій можливо автоматизувати різні сфери професійної діяльності, що створена німецькою компанією SAP AG. Найбільш популярним програмним модулем системи е ERP. Основне призначення системи - забезпечувати безперервну, взаємопов'язану, комплексну автоматизацію всіх блоків, функціональних областей і підрозділів компанії. На основі даного ПЗ в компанії створюється єдиний інформаційний простір, який структурно розподіллено по ієрархічних рівнях і за сферами роботи - збут, закупівлі, виробниче планування, оперативна діяльність, фінансовий і складський облік і так далі. Систему можна доповнювати новими модулями, інтегрувати з рішеннями сторонніх розробників. Це гнучкий і ефективний інструмент планування і управління ресурсами підприємства, за допомогою якого вона зможе вийти на якісно новий рівень результативності [11, 12].

Microsoft Dynamics. Програмний комплекс продуктів програмного забезпечення для бізнесу від компанії Microsoft. Даний комплекс допомагає збільшити результати та оптимізувати процеси за допомогою прогнозів ШІ-аналітики. Microsoft Dynamics ERP - це сімейство продуктів для планування ресурсів підприємства, орієнтованих, в першу чергу, на середній бізнес з простою корпоративною структурою і виробничою системою низької та середньої складності. Microsoft Dynamics ERP надає кошти для управління організацією (ланцюжки поставок, за купівель та управління персоналом, фінанси, проєкти спільної роботи і ін.) [13].

Під час роботи були досліджені деякі реалізації програмних продуктів для автоматизування різних сфер діяльності. В більшості продуктів,

представлені програмні рішення що базуються на ОС Windows. В деяких з них є підтримка веб-додатків у браузері. В кожної з них є свої переваги та недоліки. Проаналізувавши роботу кожної з них можна зробити висновок, що для того щоб повністю задовольнити потреби користувача, потрібно щось змінити або добавити.

Розглянемо основні недоліки готових рішень:

- відсутня підтримка мобільної веб-адаптації під системи android або ios;
- система працює повільно;
- дуже складне управління системи;
- складна конфігурація та створення системи.

1.2 Аналіз вимог до програмного забезпечення

1.2.1 Розроблення функціональних вимог

Веб-додаток має дві ролі, керівник та працівник. У ході аналізу були виявлені такі необхідні користувачу варіанти використання:

- відслідковування операцій;
- відслідковування товарів у вендингу та на складі;
- управління товарами;
- управління магазинами або точками;
- управління вендинговими автоматами;
- повідомлення працівнику про закінчення товарів у вендинговому автоматі на електронну пошту;
- аналіз даних та створення статистики за певні періоди часу.

Варіанти використання зображені у кресленику ІАЗ61.050БАК.005 Д1. Функціональні вимоги до створення веб-додатку зазначені у таблиці 1.1.

Таблиця 1.1 – Функціональні вимоги

Варіант Використання	Функціональна вимога	Пріоритет
Відслідковування операцій	1. Веб-додаток має надавати можливість перегляду операцій за певний період часу	Середній
Відслідковування товарів у вендингу та на складі	2. Веб-додаток має надавати можливість змінювати кількість товарів на складі або у вендингу 2.1 Веб-додаток має надавати можливість редагування кількість товарів на складі або вендингу 2.2 Веб-додаток має надавати можливість додавати нові товари 2.3 Веб-додаток має надавати можливість видаляти товари	Високий
Управління товарами	3. Веб-додаток має надавати можливість видаляти товари та додати нові 3.1 Веб-додаток має надати можливість редагувати товар 3.2 Веб-додаток має відображати вміст всіх товарів	Високий
Управління магазинами або точками	4. Веб-додаток має надавати можливість видаляти магазини або точки та додати нові 4.1 Веб-додаток має надати можливість редагувати магазини або точки 4.2 Веб-додаток має відображати вміст всіх магазинів або точок	Високий

Продовження таблиці 1.1.

	4.3 Веб-додаток має надавати можливість додавати або видаляти працівників які обслуговують дану ділянку	
Управління вендинговими автоматами	5. Веб-додаток має надавати можливість видаляти вендингові автомати та додати нові 5.1 Веб-додаток має надати можливість редагувати вендингові автомати 5.2 Веб-додаток має відображати вміст всіх вендингових автоматів	Високий
Повідомлення працівнику про закінчення товарів в вендинговому автоматі на електрону пошту	6. Веб-додаток повинен відслідковувати закінчення товарів в вендингу, та надсилати електронного листа з інформацією про місце знаходження автомату, товару, та кількість на даний момент	Середній
Аналіз даних та створення статистики за певні періоди часу	7. Веб-додаток має надавати можливість переглянути певну статистику за певні проміжки часу	Високий

1.2.2 Розроблення не функціональних вимог

На основі недоліків, наведених у розділі 1.1.2, з урахуванням сучасних вимог та практик, було сформовано нефункціональні вимогу:

– веб-додаток повинен підтримувати усі сучасні браузери;
– веб-додаток має коректно працювати на всіх платформах;
– інтуїтивно зрозумілий UX/UI;
– дані повинні буди захищені.

1.3 Постановка комплексної задачі розробки

Метою даної роботи є значно полегшити роботу керівникам та працівникам контролювати вміст та обіг товарів. У даній системі контролю операцій буде збережено інформацію, товарів, вендингу, локацій та зв’язки між локацію та працівником, а також які товари містить вендинговий автомат, та які операції були виконані в певній локації або вендингу. Що повинно значно полегшити роботу підприємства, в аналізі операцій та обліком товарів. Працівник підприємства в свою чергу має змогу легко відстежити, де і коли потрібна доставка товарів до вендингу, там які товари доступні на складі.

Для реалізації даної роботи повинні бути вирішені наступні задачі:

– створення архітектури клієнт-серверного програмного забезпечення;
– створення програмного-інтерфейсу взаємодії с базою даних. Інтерфейс надає можливість створення вибірки по критеріям, збереження, оновлення на видалення даних;
– створення програмного-інтерфейсу прийому платежів, які були виконані у вендингу;
– створення серверного рішення для отримання статистки. Для подальшого відображення на інтерфейсі користувача;

– створення user friendly інтерфейсу користувача для задоволення функціональних вимог. Інтерфейс повинен бути зрозумілим і легким у використанні.

1.4 Висновки до розділу

У ході аналізу вимог до розробки, було детально розглянуті та проаналізовані предметне середовище - системи контролю операцій та сучасні і доступні технології для розробки веб-додатку, розглянуті програмні продукти (1С Підприємство, SAP ERP, Microsoft Dynamics). Під час аналізу програмних продуктів, описані вище, з'ясувалось, що продукти вимагають створення окремої системи контролю. Тому було вирішено створити таку систему контроля операцій та товарів, яка б допомогла як керівництву, так і персоналу.

2 МОДЕЛЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Нижче описані загальні принципи, за якими має працювати додаток:

а) перше використання;

1) створити акаунт та видати роль керівника підприємства;
2) вивести базову статистику підприємства.

б) кожне наступне використання;

1) сформувати нову статистику;
2) сформувати новий звіт товарів у вендингу та складах.

в) розгорнути групу «User Pool» та натиснути на «Users» в боковому меню;

1) вибрати з бази даних усіх користувачів, відсортувавши за ід користувача.

г) розгорнути групу «Commerce» та натиснути «Stores» в боковому меню;

1) додати новий магазинів або точок;
2) видалення магазинів або точок;
3) редагування магазин або точку;
4) пошук по назві, або локації магазини або точки.

д) розгорнути групу «Commerce» та натиснути «Machines» в боковому меню;

1) додати нову вендингову машину;
2) видалення вендингу;
3) редагування вендингу;
4) пошук по назві або локації вегдінгу у магазині та за самою назвою магазина.

е) розгорнути групу «Commerce» та натиснути «Products» в боковому меню;

1) додавання нових товарів;
2) видалення товарів;

3) редагування товарів;

4) пошук по назві, коду або ід товару.

ж) розгорнути групу «Dashboards» та натиснути «Analytics» в боковому меню;

1) проаналізувати та відобразити статистику операцій; проаналізувати та відобразити статистику активність операцій;

2) проаналізувати та відобразити статистику товарів куплених у магазинах або точках;

3) проаналізувати та відобразити статистику товарів.

з) розгорнути групу «Dashboards» та натиснути «Project» в боковому меню;

1) проаналізувати та відобразити звіт товарів у вендингових автоматах.

За наведеними вище принципам роботи додатку, основні бізнес-процеси відповідають принципам роботи веб-додатку, які були наведені у попередньому підрозділі, та включають в себе додавання, редагування, видалення товарів, магазинів, вендингових автоматів, сторінку для статистики операцій та аналіз товарів що закінчуються.

Схеми структурні бізнес-процесів, представлені у вигляді UML нотації, наведені у кресленику ІАЗ61.050БАК.005 Д5.

2.2 Опис модулів та моделей програмного забезпечення

Застосунок складається з багатьох модулів, що відповідають за окремі функціональні частини роботи – Design Module, User Interface Module, Pages Module, Login Module, Register Module, Forgot Password Module, Mail Confirm Module, Apps Module, Project Dashboard Module, Analytics Dashboard Module, Vending Module, User Pool Module. Опис спроектованих модулів та класів наведено у таблиці 2.1.

Таблиця 2.1 – Опис основних класів веб-застосунку

Модуль	Клас	Призначення
Design Module	FuseModule	Відповідає за роботу з дизайн системою
	AppModule	Відповідає за роботу веб-додатка
	AppComponent	Компонент, який відповідає за ініціалізацію головної сторінки
	APIService	Сервіс який відповідає за роботу з базою даних
	PagesModule	Відповідає за периферійні сторінки веб-додатку
	LoginModule	Відповідає за роботу та інфраструктуру сторінки авторизації
Login Module	LoginComponent	Компонент, який відповідає за сторінку авторизації
Pages Module	RegisterModule	Відповідає за роботу та інфраструктуру сторінки реєстрації
Register Module	RegisterComponent	Компонент, який відповідає за сторінку реєстрації
Pages Module	ForgotPasswordModule	Відповідає за роботу та інфраструктуру сторінки відновити пароль
Forgot Password Module	ForgotPasswordComponent	Компонент, який відповідає за сторінку відновлення пароля

Продовження таблиці 2.1

Pages Module	MailConfirmModule	Відповідає за роботу та інфраструктуру сторінки підтвердження пошти
Mail Confirm Module	MailConfirmComponent	Компонент, який відповідає за сторінку підтвердження пошти
Apps Module	AppsModule	Відповідає за основні сторінки веб-додатку після авторизації
Apps Module	AnalyticsDashboardModule	Відповідає за роботу та інфраструктуру сторінки аналітики
Apps Module	ProjectDashboardModule	Відповідає за роботу та інфраструктуру сторінки аналізу товарів у вендингу
Apps Module	VendingModule	Відповідає за роботу комерційних сторінок
Project Dashboard Module	ProjectDashboardModule	Відповідає за роботу та інфраструктуру сторінки аналізу товарів у вендингу
Project Dashboard Module	ProjectDashboardComponent	Компонент, який відповідає за сторінку аналізу товарів у вендингу
Project Dashboard Module	ProjectDashboardService	Сервіс, який відповідає за постачання даних для компонента

Продовження таблиці 2.1

Analytics Dashboard Module	AnalyticsDashboard Module	Відповідає за роботу та інфраструктуру сторінки аналітики
Analytics Dashboard Module	AnalyticsDashboardComponent	Відповідає за сторінку аналітики
Analytics Dashboard Module	AnalyticsDashboardService	Відповідає за постачання даних аналітики
Vending Module	VendingModule	Відповідає за роботу комерційних сторінок
Vending Module	VendingMachineComponent	Компонент, який відповідає за сторінку створення або редагування вендингового автомату
Vending Module	VendingMachineService	Сервіс, що відповідає за постачання даних до компоненту
Vending Module	Machine	Модель об’єкту вендингу
Vending Module	VendingMachinesComponent	Компонент по відображенню списку вендингових автоматів
Vending Module	VendingMachinesService	Сервіс, що відповідає за постачання та дії з списком вендингових автоматів

Продовження таблиці 2.1

Vending Module	VendingProductComponent	Компонент, який відповідає за редагування або створення товарів
Vending Module	VendingProductService	Сервіс, що відповідає за постачання даних та функцій оновлення та створення товарів
Vending Module	Product	Модель об’єкту товару
Vending Module	VendingProductsComponent	Компонент по відображенню списку товарів
Vending Module	VendingProductsService	Сервіс, що відповідає за постачання та дії з списком товарів
Vending Module	VendingStoreComponent	Компонент, який відповідає за редагування або створення магазинів
Vending Module	VendingStoreService	Сервіс, що відповідає за постачання даних та функцій оновлення та створення магазинів
Vending Module	Store	Модель об’єкту магазина
Vending Module	VendingStoresComponent	Компонент по відображенню списку магазинів
Vending Module	VendingStoresService	Сервіс, що відповідає за постачання та дії з списком магазинів

Продовження таблиці 2.1

Vending Module	Product	Модель товару
Vending Module	VendingProductsComponent	Компонент по відображенню списку товарів
User Pool Module	UserPoolModule	Відповідаю за роботу та інфраструктуру для компонентів да сервіс
User Pool Module	UserPoolUserComponent	Компонент, який відповідає за оновлення працівників
User Pool Module	UserPoolUsersComponent	Компонент по відображенню списку працівників
User Pool Module	User	Модель об’єкту працівника
User Pool Module	UserPoolUserService	Сервіс, що відповідає за постачання даних об працівнику
User Pool Module	UserPoolUsersService	Сервіс, що відповідає за постачання та дії з списком працівників

Таблиця 2.2 – Опис моделі Store

Поле	Тип	Primary Key	Опис
id	Int	+	Унікальний ідентифікатор
name	String	-	Ім’я магазину або точки в системі
description	String	-	Короткий опис магазину

Продовження таблиці 2.2

location_type	String	-	Тип локації магазину або точки
location	String	-	GPS координати або адреса магазину або точки
createdAt	Date	-	Дата та час створення
updatedAt	Date	-	Дата та час оновлення
enabled	Boolean	-	Активність
machines	Array<Machine>	-	Список вендингових автоматів у магазині або точці
employees	Array<StoreEmployee>	-	Список працівників, які відповідальні за дану точку або магазин

Таблиця 2.3 – Опис моделі Machine

Поле	Тип	Primary Key	Опис
id	Int	+	Унікальний ідентифікатор
name	String	-	Ім’я вендингової машини в системі
description	String	-	Короткий опис вендингу
image	String	-	Зображення вендингового автомату
location	String	-	Точка в якій розміщений автомат у магазину або точці

Продовження таблиці 2.3

createdAt	Date	-	Дата та час створення
updatedAt	Date	-	Дата та час оновлення
store	Store	-	Магазин або точка, де встановлений вендинговій машині
products	Array<MachineProducts>	-	Список продуктів та їх кількість у вендинговій машині

Таблиця 2.4 – Опис моделі Product

Поле	Тип	Primary Key	Опис
id	Int	+	Унікальний ідентифікатор
code	String	-	Код продукту
name	String	-	Ім’я продукту в системі
description	String	-	Короткий опис товару
cost	Float	-	Ціна товару
quantity	Int	-	Кількість товарів на складі
createdAt	Date	-	Дата та час створення
updatedAt	Date	-	Дата та час оновлення
enabled	Boolean	-	Активність продукту
machines	Array<MachineProducts>	-	Список автоматів з цією модулю

Таблиця 2.5 – Опис моделі для зв’язку MachineProducts

Поле	Тип	Primary Key	Опис
id	Int	+	Унікальний ідентифікатор
machineId	Int	-	Ідентифікатор вендингового автомата
productId	Int	-	Ідентифікатор продукту
quantity	Int	-	Кількість товарів у вендинзі

Таблиця 2.6 – Опис моделі Employee

Поле	Тип	Primary Key	Опис
id	Int	+	Унікальний ідентифікатор
name	String	-	Ім’я працівника
enabled	Boolean	-	Активність працівника
stores	Array<StoreEmployee>	-	Список магазинів працівника

Таблиця 2.7 – Опис моделі зв’язку StoreEmployee

Поле	Тип	Primary Key	Опис
id	Int	+	Унікальний ідентифікатор
storeId	Int	-	Ідентифікатор магазину або точки
employeeId	Int	-	Ідентифікатор працівника

Таблиця 2.8 – Опис моделі Payment

Поле	Тип	Primary Key	Опис
id	Int	+	Унікальний ідентифікатор
storeId	Int	-	Ідентифікатор магазину або точки
machineId	Int	-	Ідентифікатор вендингу
productId	Int	-	Ідентифікатор продукту
cost	Float	-	Ціна товару на момент операції
updatedAt	Date	-	Дата та час оновлення

2.3 Висновки до розділу

У ході моделювання програмного забезпечення, було детально розглянуто та проаналізовано загальні принципи, за якими має працювати веб-додаток. Застосунок складається з багатомодульної архітектури, що відповідають за необхідні частини роботи веб-додатка. Таким чином було описано усі базові модулі та їх вміст, а також моделі класів бази даних.

3 ОПИС ПРОГРАМНОГО ТА ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз відомих технічних рішень

За останні 5 років з’явилося багато JavaScript фреймворків та бібліотек призначені для створення одно-сторінкових (SPA) веб-додатків. Найпопулярнішим патернов даних фреймворків є MVC. Як приклад такими рішеннями є Angular, AngularJS, React, Vue та Next.JS. Серед яких React бібліотека на даний час, являється лідером у цій сфері, але Angular та Vue.JS теж популярними рішеннями зі стабільною архітектурою самого фреймворку. Про це свідчить порівняння запитів у системі Google Trends приведена на рисунку 3.1.

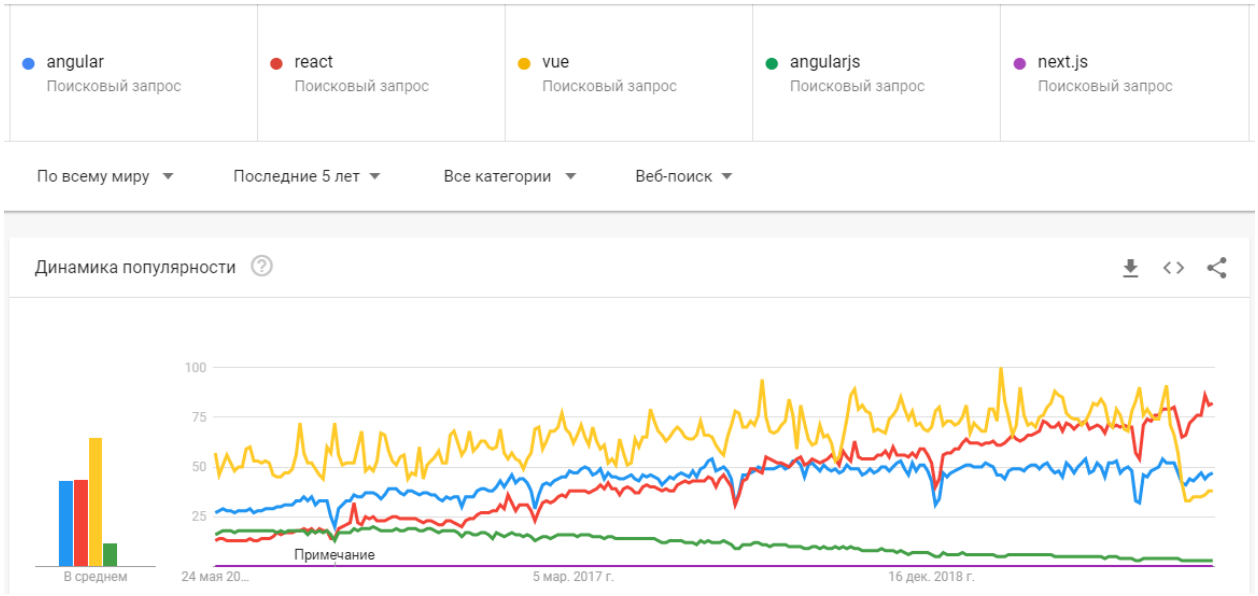

Рисунок 3.1 – Статистика сервісу Google Trend в порівнянні тенденцій фреймворків [27]

На даний час існує багато технічних рішень для створення інтерфейсу користувача. Розглянемо основні з них.

1. React. Бібліотека JavaScript, призначена для створення інтерфейсів користувача. React буд створений компанією Facebook у 2013 році та продовжує розробку разом з спільнотою. Спочатку бібліотека призначалася тільки для створення веб-сайту, проте пізніше з'явилася платформа React Navite, яка все призначається для створення додатків для мобільних пристроїв. Бібліотека дозволяє розробникам створювати масштабовані веб-додатки, які

використовують дані, котрі змінюються з часом без перезавантаження сторінки. Його мета полягає в тому, щоб бути простим, масштабованим, швидким. Сама сторінка складається як з самого коду так і стилів та HTML, які включені у єдиний файл. З використанням бібліотеки інтерфейсів React можуть бути використана бібліотека Redux для обробки та зберігання станів веб-додатку. Уся структура веб-сторінки може бути представлена у вигляді DOM (Document Object Model) - організація елементів html, якими ми можемо маніпулювати, змінювати, видаляти або додавати нові. Для взаємодії з DOM застосовується мова JavaScript. Однак коли намагаємося маніпулювати html-елементами за допомогою JavaScript, то можемо зіткнутися зі зниженням продуктивності, особливо при зміні великої кількості елементів. А операції над елементами можуть зайняти деякий час, що неминуче позначиться на призначеному для користувача досвід. Однак якби працювали з коду js з об'єктами JavaScript, то операції проводилися б швидше. Для вирішення проблеми продуктивності якраз і з'явилася концепція віртуального DOM. Віртуальний DOM представляє легковажну копію звичайного DOM. І відмінною рисою React є те, що дана бібліотека працює саме з віртуальним DOM, а не звичайним. Якщо з додатком потрібно дізнатися інформацію про стан елементів, то відбувається звернення до віртуального DOM. Якщо необхідно змінити елементи веб-сторінки, то зміни спочатку вносяться в віртуальний DOM. Потім новий стан віртуального DOM порівнюється з поточним станом. І якщо ці стани розрізняються, то React знаходить мінімальну кількість маніпуляцій, які необхідні до поновлення реального DOM до нового стану і виробляє їх. У підсумку така схема взаємодії з елементами веб-сторінки працює набагато швидше і ефективніше, ніж якби ми працювали з JavaScript з DOM безпосередньо [2, 3].

Переваги:

– легко вивчити, завдяки простому дизайну, використання JSX (HTML-подібний синтаксис) для шаблонів і дуже докладної документації. Розробники витрачають більше часу на написання сучасного JavaScript і менше турбуються про код, специфічному для фреймворка;

– дуже швидка, завдяки реалізації React Virtual DOM і різним оптимізаціям рендеринга;

– відмінна підтримка рендеринга на стороні сервера, що робить його потужною платформою для контент-орієнтованих додатків;

– першокласна підтримка Progressive Web App (PWA) завдяки генератору додатків `create-react-app`;

– прив'язка даних є односторонньою, що означає менше небажаних побічних ефектів;

– redux, найпопулярніша платформа для управління станом додатків в React, її легко вчити і використовувати;

– react реалізує концепції функціонального програмування (FP), створюючи простий в тестуванні і багаторазово використовуваний код;

– додатки можуть бути створені за допомогою TypeScript або Facebook's Flow, що мають вбудовану підтримку JSX;

– перехід між версіями, як правило, дуже простий: Facebook надає «кодові модулі» для автоматизації більшої частини процесу;

– навички, отримані в React, можуть бути застосовані до розробки на React Native.

Недоліки:

– react не однозначний і залишає розробникам можливість вибирати кращий спосіб розвитку. Це може бути вирішено сильним лідерством проєкту і хорошими процесами;

– спільнота ділиться по способам написання CSS в React, які поділяються на традиційні таблиці стилів (CSS Modules) і CSS-in-JS (тобто Emotion і Styled Components);

– react відходить від компонентів на основі класів, що може стати перешкодою для розробників, яким більш комфортно працювати з об'єктно-орієнтованим програмуванням (ООП);

– змішування шаблонів з логікою (JSX) може збити з пантелику деяких розробників при перших знайомствах з React.

Компанії, які використовують React: Facebook, Instagram, Netflix, New York Times, Yahoo, Khan Academy, Whatsapp, Codecademy, Dropbox, Airbnb, Asana, Atlassian, Intercom, Microsoft, Slack, Storybook і багато інших [25].

2. Angular. Framework з Open Source кодом, який розробляється Google та спільнотою. Перш за все він націлений на розробку SPA-рішень, що складаються з компонентів сторінок, в які входять HTML шаблон з SCSS і TypeScript. Однією з ключових особливостей Angular є те, що він використовує в якості мови програмування TypeScript. Головна мета фреймворку розробка браузерних веб-додатків на основі партерну MVC. За допомогою інтегрованих інших фреймворків це полегшує тестування додатку. Як і інші фреймворки, цей працює на основі DOM елементів. Головною сторінкою завжди є компонент в якому знаходиться шаблони, стилі та програмний код а так же допомагає скорегувати тип рендирингу компонента, вручну або автоматично при зміні програмної моделі сторінки. Фреймворк працює на так званій системі біндингів, що допомагає закріпити на шаблоні програмні моделі [5, 6].

Переваги:

– angular використовується разом з TypeScript. Він має виняткову підтримку для цього;

– angular-language-service - забезпечує інтелектуальні можливості і автозаповнення шаблону HTML-компонента;

– нові функції, такі як generation Angular, що використовують бібліотеки npm з CLI, generation, і розробка компонентів, що використовує Angular;

– детальна документація, що дозволяє розробнику отримати всю необхідну інформацію, не вдаючись до допомоги його колег. Однак це вимагає більше часу для навчання;

– одностороння прив'язка даних, яка забезпечує виняткову поведінку додатка, що зводить до мінімуму ризик можливих помилок;

– mvvm (Model-View-ViewModel), яка дозволяє розробникам працювати окремо над одним і тим же розділом програми, використовуючи один і той же набір даних;

– впровадження залежностей від компонентів, пов'язаних з модулями і модульність в цілому;

– структура і архітектура, спеціально створені для великої масштабованості проєкту.

Недоліки:

– різноманітність різних структур (Injectables, Components, Pipes, Modules і т. Д.) Ускладнює вивчення в порівнянні з React і Vue.js, у яких є тільки «Component»;

– відносно повільна продуктивність, враховуючи різні показники. З іншого боку, це можна легко вирішити, використовуючи так званий «ChangeDetectionStrategy», який допомагає вручну контролювати процес рендеринга компонентів.

Компанії, які використовують Angular: Microsoft, Autodesk, MacDonald's, UPS, Cisco Solution Partner Program, AT & T, Apple, Adobe, GoPro, ProtonMail, Clarity Design System, Upwork, Freelancer, Udemy, YouTube, Paypal, Nike, Google, Telegram, Weather, iStockphoto, AWS, Crunchbase [25].

3. Vue.js. Open Source Framework який базується на JavaScript. Даний фреймворк призначений для створення веб-додатків клієнтського рівня. Основна сфера застосування даного фреймворка - це створення і організація призначеного для користувача інтерфейсу. Його творцем є Еван Ю (Evan You), який до цього працював в Google над AngularJS. Vue.js використовує віртуальний DOM. Віртуальний DOM представляє більш продуктивну копію реального DOM. Для створення сторінки використовується синтаксис шаблонів на базі HTML, що дозволяє декларативно зв'язати між віртуальним DOM на контекстом даних у Vue, або крім самих шаблонів, фреймворк дозволяє написати рендеринг функцію використовуючи JSX. Якщо дані, які використовуються у контекстному середовищі Vue.js, змінюються, то зміни

спочатку вносяться в віртуальний DOM. Потім Vue вибирає мінімальний набір компонентів, для яких треба виконати зміни на веб-сторінці, щоб реальний DOM відповідав віртуальному. Завдяки віртуальному DOM підвищується продуктивність програми [7, 8].

Переваги:

– посилений HTML. Це означає, що Vue.js має багато характеристик схожих з Angular, а це, завдяки використанню різних компонентів, допомагає оптимізації HTML- блоків;

– детальна документація. Vue.js має дуже детальну документацію, яка може прискорити процес навчання для розробників і заощадити багато часу на розробку програми, використовуючи тільки базові знання HTML і JavaScript;

– адаптивність. Може бути здійснений швидкий перехід від інших фреймворків до Vue.js через схожість з Angular і React з точки зору дизайну і архітектури;

– приголомшлива інтеграція. Vue.js можна використовувати як для створення односторінкових додатків, так і для більш складних веб-інтерфейсів додатків. Важливо, що невеликі інтерактивні елементи можна легко інтегрувати в існуючу інфраструктуру без негативних наслідків;

– масштабування. Vue.js може допомогти в розробці досить великих шаблонів багаторазового використання, які можуть бути зроблені майже за той же час, що і більш прості;

– крихітний розмір. Vue.js важить близько 20 КБ, зберігаючи при цьому свою швидкість і гнучкість, що дозволяє досягти набагато кращої продуктивності в порівнянні з іншими платформами.

Недоліки:

– недолік ресурсів. Vue.js як і раніше займає досить невелику частку ринку в порівнянні з React або Angular, що означає, що обмін знаннями в цьому середовищі все ще перебуває на початковій стадії;

– ризик надмірної гнучкості. Іноді у Vue.js можуть виникнути проблеми при інтеграції в величезні проєкти, і поки ще немає досвіду можливих рішень, але вони обов'язково з'являться найближчим часом.

Компанії, які використовують Vue.js: Xiaomi, Alibaba, WizzAir, EuroNews, Grammarly, Gitlab і Laracasts, Adobe, Behance, Codeship, Reuters [25].

3.2 Аналіз відомих хмарних рішень

3.2.1 Огляд хмарних технологій

Хмарні технології є доволі популярними в наш час, легкими в використанні та найголовніше доступ до власних даних, які зберігаються в хмарі та можна користуватися з будь-якого пристрою.

На сьогоднішній день є доволі багато хмарних технологій, які використовують хмарні обчислення. Хмарні обчислення (англ. Cloud Computing) надалі "хмара" – це доступ на замовлення ресурсів комп'ютерної системи, особливо для зберігання даних (хмарного сховища) та обчислювальної потужності, та на пряму активного управління користувачем. Цей термін зазвичай використовується для опису центрів обробки даних, доступних багатьом користувачам через мережу Інтернет. На сьогоднішній день зазвичай використовувати великі хмари, які мають функції розподілення по декількох місцях від центральних серверів. В такому разі з’єднання залежить від відстані, і користувач приєднується до найближчих серверів.

Мета хмарних обчислень – дозволити користувачам без зусиль скористатись усіма технологіями, які не потребують спеціальних знань чи досвіду роботи з кожною з них. Робота з хмарами дозволяє значно скоротити витрати і цей інструмент допомагає користувачам зосередитися на основному бізнесі, а не перешкоджати IT-сфері. Основна технологія хмарних обчислень – віртуалізація. Програмне забезпечення для віртуалізації відокремлює фізичний обчислювальний пристрій на один або декілька "віртуальних" пристроїв, кожен з яких може бути легко використовуватись та керуватись для того щоб

виконувати обчислювальні завдання. Завдяки, тому що віртуалізація відбувається на рівні операційної системи, що створює масштабовану систему з декількох незалежних обчислювальних пристроїв. Виходячи з цього простої обчислювальні ресурси можуть бути розподілені та використані на багато ефективніше. Таким чином зменшуються витрати за рахунок збільшення використання інфраструктури, віртуалізація забезпечує спритність, необхідну для прискорення IT-операцій. Автономні обчислення автоматизують процес, таким чином користувач може надавати ресурси на вимогу. Автоматизація прискорює процес, мінімізуючи залучення користувачів, зменшує затрати праці та зменшує можливість помилок людини.

Хмарні обчислення почали займає все більш і більш вагому роль в практиці, проникаючи в різні IT-сфери. Вагому роль в популяризовані технології відіграла компанія "Amazon.com", випустивши свій продукт Elastic Compute Cloud у 2006 році. Хоча перші згадки про "хмарні обчислення" з'явилися ще в 1996 році, у внутрішніх документах Compaq. Як попередники Інтернету, хмарний символ використовувався для представлення мереж обчислювальної техніки в оригінальній ARPANET ще в 1977 р. та CSNET до 1981 р.. Слово хмара використовувалося як метафора для Інтернету, а хмарна форма застосовувалась для позначення мережі на телефоні. Під даним спрощеннями маємо на увазі, що специфіка з'єднання кінцевих точок мережі не є актуальною для розуміння діаграми.

3.2.2 Сервісні моделі

На даний день існує 3 основні моделі обслуговування:

1) IaaS. Інфраструктура як послуга. Орендується виділений сервер або хмарні обчислення. Постачальник послуг гарантує лише працездатність на рівні сервера (електронна начинка сервера, інтернет, електроживлення тощо.) або технології віртуалізації.

Переваги:

– «закопувати» гроші в покупку сервера не завжди розумно, тому що він з часом застаріває. Орендуючи сервер ви завжди можете його поміняти чи обновити з мінімальними витратами;

– можна вибирати і керувати продуктивністю і параметрами сервера як хочеться;

– повна свобода в реалізації бажаного. Можна встановлювати будь-які операційні системи, програми. Настроювання та використання як заманеться;

– гарне співвідношення ціна/можливості.

Недоліки:

– необхідний свій фахівець для настройки та обслуговування програмного забезпечення;

– часто є прив'язка до конкретному характеристикам сервера або технології віртуалізації і трохи втрачається гнучкість.

2) PaaS. Платформа як послуга. Орендується хмарні обчислення, сховища, трафік та інше, які консолідуються у деяку платформу.

Переваги:

– приголомшлива гнучкість - можна зібрати комп'ютер будь-якої потужності (від мікро сервера, який можна порівняти за потужністю з смартфоном до кластера з сотень тисяч серверів), і встановити на нього самі різні програми;

– основні постачальники мають величезні територіально розподілені обчислювальні мережі, що дозволяє легко розгортати швидкі, масові і відмовостійкі додатки;

– є можливість додатково підключити просунуті послуги, в яких у Oracle, Microsoft, Google є унікальні компетенції - використання штучного інтелекту, аналіз великих обсягів інформації і т.д.;

– є можливість оплачувати тільки фактично спожиті ресурси.

Недоліки:

– для використання потрібно «зібрати» систему з віртуальних компонентів-складових. Це вимагає специфічних знань і умінь;

– кожна платформа накладає свої обмеження при реалізації, повної свободи в реалізації немає;

– досить висока базова вартість.

3) SaaS. Програмне забезпечення як послуга. Самий популярний для користування хмарою. Усі необхідні функції поступні через Інтернет.

Переваги:

– технічні деталі повністю приховані. Підключаємося через інтернет і користуємося готовою програмою. Оновлення, працездатність та інші технічні питання і проблеми вирішує постачальник послуг;

– як правило мінімальні терміни доступності. Досить замовити послугу і можна користуватися через кілька хвилин.

Недоліки:

– не всі програми бувають доступні в такому форматі з технічних причин;

– можливості налаштування і зміни програми під свої вимоги обмежені;

– сильна залежність від постачальника послуги і якості його роботи;

– часто ви маєте обмежений контроль над своїми власними даними;

– часто виходить дорожче інших типів хмар або класичних додатків, особливо для великих замовників.

За останні 5 років, компанія Amazon стала дуже популярна у області хмарних обчислень. Компанія базується на PaaS моделі. За статистикою Google Trends, платформа AWS (Amazon Web Services) є лідером у даній сфері, рисунок 3.2.

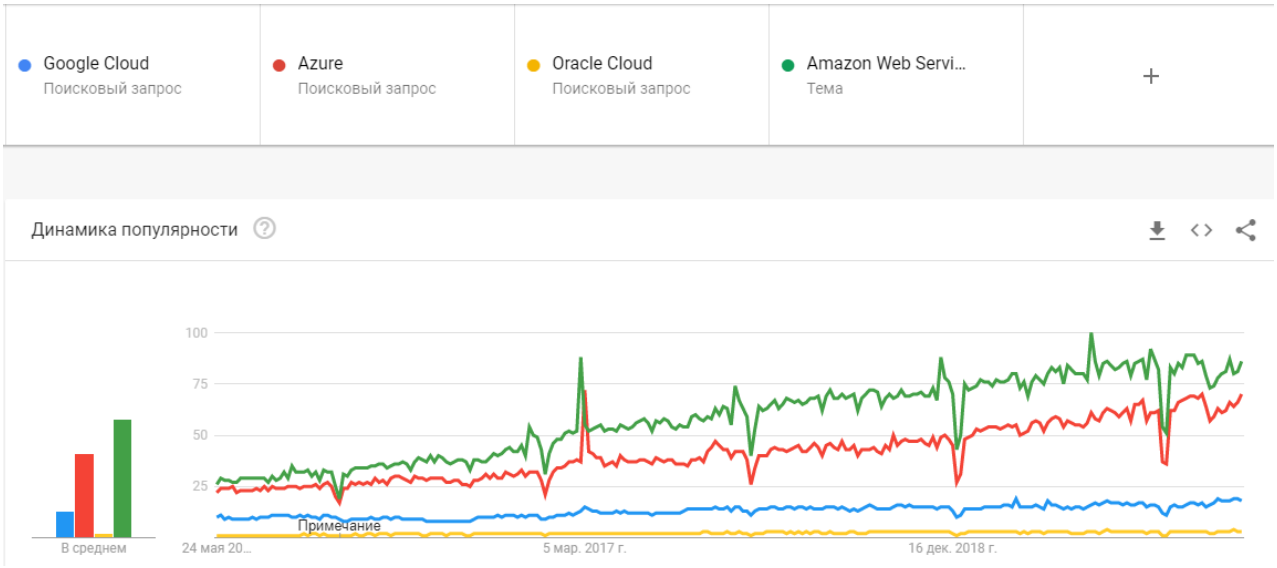

Рисунок 3.2 – Статистика популярності хмарних рішень [28]

3.2.3 Аналіз існуючих рішень

Існую багато хмарних рішень, але ми розглянемо най популярніші рішення, за статистикою Google Trend, рисунок 3.2.

Amazon Web Services (AWS). комерційна публічна хмара, яка підтримується і розвивається компанією Amazon з 2006 року. Надає передплатникам послуги по двом направленням, як по інфраструктурної моделі (віртуальні сервери, ресурси зберігання), так і платформного рівня (хмарні бази даних, хмарне сполучна програмне забезпечення, хмарні несерверні обчислення, засоби розробки). В хмарі є різноманітні хмарні СУБД різних категорій. Серед доступних NoSQL-систем - Amazon SimpleDB, DynamoDB, резидентная СУБД ElastiCache, графовая СУБД Neptune. В рамках послуг Amazon Relational Database Service (RDS) передплатники можуть розгортати хмарні бази під управлінням популярних реляційних СУБД - MySQL, Oracle Database, Microsoft SQL Server і PostgreSQL, при цьому також доступна масштабируемая реляційна СУБД Amazon Aurora, сумісна з MySQL і PostgreSQL. Аналітична масово-паралельна реляційна СУБД ParAccel, адаптована для хмарної інфраструктури, надається під торговою маркою Amazon Redshift [14].

Microsoft Azure. хмарна платформа компанії Microsoft. Надає можливість розробки, виконання додатків і зберігання даних на серверах, які розташовані в розподілених дата-центрах. Хмара Azure було анонсовано в жовтні 2008 року під кодовою назвою "Project Red Dog". Реліз відбувся 1 лютого 2010 під уже іншою назвою "Windows Azure". У 2014 році у платформа була змінена назва на Microsoft Azure. Microsoft Azure реалізує хмарні моделі платформи як сервісу (PaaS) та інфраструктури як сервісу (IaaS). Можливе використання як сторонніх, так і сервісів Microsoft в якості моделі ПО як сервісу (SaaS). Працездатність платформи Microsoft Azure забезпечує глобальна мережа розподілених дата-центрів Microsoft. Крім стандартних функцій операційних систем, Microsoft Azure має і додаткові: виділення ресурсів на вимогу для масштабування, автоматичне синхронну реплікацію даних для підвищення відмово стійкості, обробку відмов інфраструктури для забезпечення постійної доступності та інше. Модель надання інфраструктури (IaaS) надає можливість оренди таких ресурсів, як сервери, пристрої зберігання даних та мережеве обладнання. Управління всією інфраструктурою відбувається постачальником, споживач тільки управляє операційною системою і встановленими додатками. Для віртуальних машин доступні образи наступних операційних систем: Windows Server, CoreOS, Ubuntu Server, Red Hat, Clear Linux OS, Debian, SUSE Linux Enterprise Server, Oracle Linux. Практично всі сервіси Microsoft Azure мають інтерфейс взаємодії API, побудований на системі обмежень для розподілених систем REST, що дозволяє користувачам використовувати хмарні сервіси з будь-якою операційною системи, пристрої і платформи. Крім того, розробники можуть створювати і управляти власними сервісами, користуючись візуальним веб-інтерфейсом порталу Azure. Портал надає можливість налаштовувати сервіси, редагувати права доступу, відстежувати стан ресурсів і управляти біллінгом [4, 15].

3.3 Засоби розробки

Для розробки веб-додатку було використано фреймворк Angular, який базується на мові TypeScript. Вона представляє мову з суровою типізацією. Додаток компілюється у JavaScript. Можливості TypeScript необхідні для реалізації:

- систему роботи с класами;
- наслідування від інших інтерфейсів та класів;
- опис власних типів даних;
- опис сигнатур методів;
- опис типів змінних.

Для створення повноцінного веб-додатку використовується платформа AWS Amplify, яка надає такі можливості:

- створення ламбда функцій;
- створення API Endpoint’s;
- створення медія-сховищ та баз даних;
- генерація моделей GraphQL;
- надання системи інтеграції з ідентифікації користувачів.

3.4 Архітектура програмного забезпечення

Веб-додаток повинен містити 5 головних модулі, які відповідають за певну задачу:

- модуль дизайн системи;
- модуль аналітики даних;
- модуль роботи з користувачами;
- модуль комерції, який включає в себе функції товарів, магазинів або точок та вендингових автоматів;
- модулі сторінок авторизації.

– зі свого боку серверна архітектура повинна містити такі сервіси:
– сервіс прийому операцій від вендингових автоматів;
– сервіс роботи з користувачами;
– сервіс аналітики.

3.4.1 Загальні відомості про REST API

REST (Representational State Transfer - «передача стану уявлення») - архітектурний стиль взаємодії компонентів розподіленого додатка в мережі. REST є узгоджений набір обмежень, що враховуються при проектуванні розподіленої гіпермедіа-системи.

REST API має на увазі під собою прості правила:

– кожен URL є ресурсом;
– при зверненні до ресурсу методом GET повертається опис цього ресурсу;
– метод POST додає новий ресурс;
– метод PUT змінює ресурс;
– метод DELETE видаляє ресурс.

Ці правила надають простий CRUD інтерфейс для інших додатків, взаємодія з яким відбувається через протокол HTTP.

Відповідність CRUD операцій і HTTP методів:

– create – POST
– read – GET
– update – PUT
– delete – DELETE

REST API інтерфейс дуже зручний для між програмної взаємодії, наприклад мобільний додаток може виступати в ролі клієнта, який маніпулює даними за допомогою REST [16, 17].

3.4.2 Загальні відомості про GraphQL

GraphQL — це мова запитів і маніпуляції даними з відкритим кодом для API і середовище виконання для обслуговування запитів з наявних даних. GraphQL надає підхід розробки веб API і його можна порівнювати і протиставляти REST та іншим протоколам зв'язку веб-сервісів. Він дозволяє клієнтам визначати структуру потрібних даних і таку саму структуру повертає сервер, таким чином запобігаючи передачі надлишкових даних, але це впливає на дієвість веб-кешування результатів запитів. Гнучкість і багато образність мови запитів, що може бути не потрібна для простих API. Він складається з системи типів, мови запитів і семантики виконання, статичної валідації і інтроспекції [18].

Отже це також дає змогу повного інтерфейсу CRUD.

3.4.3 Загальні відомості про AWS Amplify

AWS Amplify - це платформа для розробки надійних і масштабованих мобільних і інтернет-додатків. Завдяки цій платформі значно спрощуються такі завдання, як аутентифікація користувачів, надійне зберігання даних і метаданих користувача, вибіркове надання доступу до даних, інтеграція машинного навчання, аналіз метрики додатків і виконання коду на стороні сервера. Amplify охоплює весь робочий процес розробки мобільного застосування від контролю версії і тестування коду до виробничого розгортання і легко масштабується в міру розвитку бізнесу, дозволяючи збільшити кількість користувачів від тисяч до десятків мільйонів. Бібліотеки та інтерфейс командного рядка Amplify з відкритим вихідним кодом є частиною платформи Amplify. В інтерфейсі можна налаштовувати і створювати власні модулі [23].

3.4.4 Загальні відомості про Amazon Cognito

Amazon Cognito - дозволяє швидко і просто додавати можливості реєстрації, авторизації і контролю доступу користувачів в мобільні та інтернет-додатки. Amazon Cognito масштабується до мільйонів користувачів і підтримує авторизацію за допомогою соціальних постачальників посвідчень (Facebook, Google, Amazon), а також постачальників корпоративних посвідчень на основі SAML 2.0. Сервіс Amazon Cognito User Pools надає безпечний каталог користувачів, здатний масштабуватися до сотень мільйонів користувачів. User Pools - це повністю керований сервіс, тому його легко налаштувати, не створюючи спеціальну серверну інфраструктуру [24].

Сервіс підтримує стандарти підтримує стандарти управління ідентифікацією та доступом, такі як Oauth 2.0, SAML 2.0 і OpenID Connect. Amazon Cognito відповідає вимогам HIPAA і стандартам PCI DSS, SOC, ISO / IEC 27001, ISO / IEC 27017, ISO / IEC 27018 та ISO 9001. Надає рішення для контролю доступу до серверних ресурсів з програми. Це дозволяє визначати ролі, а також призначати різні ролі користувачам, щоб додаток отримувало доступ тільки до ресурсів, доступним конкретного користувача. Вбудований призначений для користувача інтерфейс і проста настройка федерації з постачальниками посвідчень дозволяють в лічені хвилини інтегрувати Amazon Cognito і додати в додаток можливості реєстрації, авторизації і контролю доступу користувачів.

3.4.5 Загальні відомості про AWS Lambda

AWS Lambda – це один з багатьох сервісів AWS (Amazon Web Services), що дозволяє запускати програмний код без виділення серверів і керування ними. Оплата підлягає тільки за фактичний час виконання обчислень.

За допомогою Lambda можна запускати практично будь-які види додатків і серверних сервісів, при цьому не потрібні будь-які операції адміністрування для цього потрібно скористатися веб-сторінкою. Просто завантажте програмний код, і Lambda забезпечить всі ресурси, необхідні для його виконання, масштабування і забезпечення високої доступності. Можна налаштувати автоматичний запуск програмного коду з інших сервісів AWS або викликати його безпосередньо з будь-якого мобільного або інтернет-додатки [19].

3.4.6 Аналіз середовища виконання програмного забезпечення

Для реалізації даної роботи, яка повинна підтримувати найпопулярніші платформи, потрібно мати хоча б одну із сучасних операційних систем, таку як Windows, Linux, MacOS, Android. На даній ОС має бути встановлений один із сучасних браузерів такі які Chrome, Firefox, Opera, Edge які повинні підтримувати JavaScript стандарту ECMAScript 5. В основі даної системи буде використовувати сучасно версію фреймворку Angular 9.1. На даний час Agnular являється одним з стабільних та економних фреймворків. Angular інтегрував в собі багато сучасних програмних підходів так бібліотек. Однією з головних тенденцій яка реалізована в фреймворці це Dependency Injection (DI), він являється важливим дизайном системи. DI — шаблон проектування програмного забезпечення, що передбачає надання зовнішньої залежності програмному компоненту, використовуючи «інверсію управління» (англ. Inversion of control, IoC) для розв'язання (отримання) залежностей [20].

В Angular система Dependency Injection забезпечує оголошені залежності в класах які реалізовуються за допомогою Angular декораторів. Для позначення класів сервісів, директив, компонентів, гвардів та інших структурних класів Angular, використовуються функції-декоратори @Component, @Directive, @Injection, @NgModule, які в свою чергу при компіляції перетворюється на

метадані. Сам веб-додаток структурно складається компонентів, сервісів, директив та інших структур Angular. Архітектура фреймворка Angular приведена на рисунку 3.3.

Компоненти відповідають за певну частину DOM-обьектів та містить шаблони, стилі та логіку, та дочірні компоненти [21]. Ієрархія та зв'язки рендерінгу шаблонів компонентів у Angular фреймворці відображена на рисунку 3.4.

Директива поділяються на структурні або атрибутивні. Компонент технічно є директивою також, але з він настільки відмінний, що Angular визначає декоратор @Component, який розширює декоратор @Directive за допомогою ООП. Структура директива виконує створення, видалення або змінює елементи DOM. Атрибутивні директиви створенні для роботи з існуючими HTML елементами, яка може відслідковувати дії існуючого шаблону або змінювати її поведінку.

Рисунок 3.3 – Архітектура фреймворка Angular [26]

Сервіс - це широка категорія, що включає будь-яке значення, функцію, яка потрібна додатку. Служба, як правило, є класом із вузькою, чітко визначеною метою. Він повинен робити щось конкретне і робити це добре. Angular відрізняє компоненти від служб для підвищення модульності та повторного використання. Відокремлюючи функціональний вигляд компонента від інших видів обробки, ви можете зробити класи компонентів піщаними та ефективними [22].

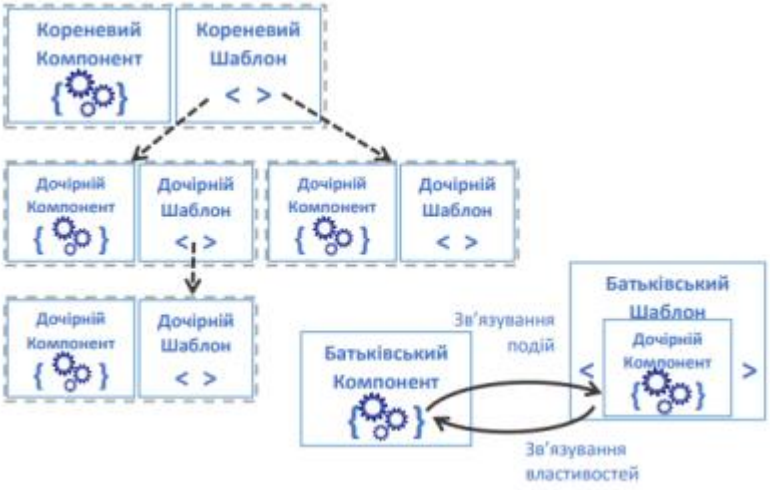

Рисунок 3.4 – Ієрархія та зв'язок рендерінгу шаблонів компонентів у фреймворці Angular [21]

Застосунок складається з 4 модулів, що відповідають за окремі функціональні частини роботи – Design Module, User Interface Module, Pages Module, Apps Module. Опис спроектованих модулів та класів наведено у таблиці 2.1. Для реалізації REST API та GraphQL API було використано сервіс AWS Lambda та AWS Gateway для створення виділеної точки доступу до інтерфейсів взаємодії з базою даних, та сервісу AWS Cognito для аутентифікації та реєстрації. Данні сервіси уже інтегровані у фреймворк Amplify. Тому для підключення даних сервісів потрібно лише сконфігурувати їх.

3.4.7 Архітектурні компоненти програмного забезпечення

Сервіс роботи з базою даних, бібліотека Amplify має вбудований інтерфейс по взаємодії с DynamoDB. Для взаємодії с базою даних спочатку створюється схема самої бази даних. На даний момент модель бази даних складається з 7 моделей - Store, Machine, MachineProducts, Product, StoreEmployee, Employee, Payment, які відповідають моделям класів та описані у таблицях 2.2 – 2.8. На основі даної моделі генерується сервіс для взаємодії з базою даних. Його опис наведений у додатку В таблиці В.1.

Модулі інтерфейсу складаються з компонентів та сервісів, що відповідають необхідним частинам поведінки веб-додатку. Опис основних методів класів наведений у додатку В таблиці В.2.

Основні використані елементи Angular:

– component – декоратор що помічає компонент, в який входять як HTML шаблони, стилі та поведінку інтерфейсу користувача;

– service (Injectable) – провайдер інтерфейсу бази з функцій обробки даних без бізнес логіки;

– layout – спеціальні контейнери розмітки

– graphql api – сервіс для взаємодії з базою даних за допомогою мови запитів, побудована на GRUD моделі. Опис моделі бази даних представлена у додатку Б;

Також основою будь-якого Angular веб-додатку є файли package.json та package-lock.json, які базуються на менеджері пакетів NPM, котрі містить важливу інформацію про веб-додаток. Модулі містять перелік усіх компонентів, сервісів, директив, на інших модулів для мінімізації залежностей пакетів, які необхідний для роботи додатку або компоненту. А package. json – всю необхідну інформацію для компілювання проєкту, в тому числі залежності використаних бібліотек. Angular.json – містить головні схеми та параметри компіляції додатка там інші важливі опції для розробки та тестування.

Для реалізації REST API було використано сервіс AWS Lambda. За допомогою фреймворку Amplify, який слідкую як за веб-додатком так мікро-серверами, було створено 3 функції на мові JavaScript у середовищі Node.JS 12.+. AWS Lambda для програмного середовища Node.JS базується на бібліотеці aws-serverless-express, що дужий схожий на фреймворк Express для обробки веб-запитів по протоколу HTTP. Але у подальшому функція у AWS Lambda викликається через API Gateway, та для полегшення подальшої обробки протоколу HTTP було додатково обрано фреймворк Express, який працює за допомогою маршрутів.

У даному проєкті буде використано взаємодія через REST API, як з сторони веб-додатку так і вендингу. Але вендингові автомати здебільшого

базуються на інших типах протоколів, такі як WebSocket, MQTT, та інші. Ці протоколи взаємодії вендинга з сервером будуть розроблені у наступній версії.

Для повного функціонування веб-додатку було використано такі бібліотеки з відкритим кодом:

– amplify – фреймворк для взаємодії з сервісами AWS.

– angular material – бібліотека що містить велику кількість візуальних компонентів для розробки клієнтських веб-додатків на мові програмування TypeScript;

– aws-serverless-express - JavaScript бібліотека для AWS Lambda Node.JS функцій;

– express – JavaScript бібліотека для зручного створення сервера на HTTP протоколі;

– axios – JavaScript бібліотека для зручної відправки HTTP запитів у ендпоінти;

– graphql – бібліотека для роботи з запитами на мові GraphQL;

– aws-sdk – JavaScript бібліотека для роботи з сервісами AWS на стороні бекенду.

При розробці даного веб-додатку, використовувалася система контролю версій GIT, для авто-деплою та перевірки середовище сервіс Amplify.

3.5 Висновки до розділу

Застосунок складається з багатомодульної архітектури, що відповідають за необхідні частини роботи веб-додатка. Таким чином веб-додаток можливо масштабувати у будь яких пропорціях по мірі необхідності або задати авто-масштабування, таким чином зростає продуктивність та відмово стійкість системи. Модулі інтерфейсу користувача складається з компонентів. Вони розроблені відповідно до типової архітектури застосунку Angular та надає користувачу зрозумілий та зручний інтерфейс для роботи з веб-додатку. А також функції для взаємодії з аналізом даних.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз якості програмного забезпечення

Тестування продукту посідає не менш важливу роль в процесі розробки та випуску продукту, адже це забезпечить стабільне функціонуванню продукту в подальшому.

Тестування ПЗ – це процес перевірки відповідності поведінки програми між реальним і очікуваним результатом. При тестуванні потрібно враховувати різні фактори: середовище в якому тестують(різні версії операційних систем; різні розміри екранів, як ПК так і мобільних пристроїв), вимоги та функції, які необхідно виконати та різні типи з’єднання до мережі та обриву зв'язку тощо. Щоб протестувати продукт інженери QA використовують різні інструменти в залежності від специфіки та специфікації. Від цього буде залежати методологія тестування.

При підтвердженні правильної роботи додатку, потрібно послідовно виконувати тест кейси, що дозволить наглядно відобразити функціональну поведінку та зручність використання, до того як випускати його на широкий загал.

Преваги тестування:

– пошук багів до публікації веб-додатку;
– виявлення несправності у циклі розробки;
– виявити технічні характеристики та оптимізувати їх.

Тестування веб-додаток має свої особливості:

– перевірка великої кількості браузерів та їх версій;
– моделі пристроїв відрізняються;
– розміри екрана різняться;
– перевірка клавіатури у мобільних ОС;
– ОС пристрою швидко старіє;
– широкий спектр конкретних операційних систем та компонентних конфігурацій;

– перевірка на різних швидкостях передачі даних.

Для того щоб отримати задовільний результат при тестуванні веб-додатку, потрібно ретельна перевірка функціональності додатку.

4.2 Опис процесу тестування

Як було зазначено в попередньому пункті, тестування веб-додатку має свої особливості. Зазвичай тестування розділяють на модульне (Unit Testing), системне (System Testing), інтеграційне (Integration Testing) тощо. Але в тестування веб-додатку виділяють на малі, середні та великі. Ці тести орієнтуються на охоплення об’єму коду.

Малі тести – це тести, які можуть перевірити одну функцію чи модуля, окремо від виробничих систем. Їх, як правило, пишуть розробники і їх виконання займає декілька секунд. Інструментом для виконання тестування використовують Karma.

Середні тести – це тести, які покривають дві чи більше функцій. Їх задача перевірити взаємодію між цими функціями, які можуть контактувати напряму чи визивати один одного.

Великі тести – це тести, які покривають не менше трьох чи більше функцій. При тестуванні використовуєте реальні сценарії користувача. Мета великих тестів перевірить чи відповідає продукт потребам користувача. Таке тестування проводять, або вручну, або автоматизують. При тестуванні користуються інструментом Karma.

При тестуванні веб-додатку, окрім основного функціонального та юзабіліті тестування, також виконують тестування параметра середовища: зміна розміру екрана, робота у фоновому режимі, оновлення сторінки, надання/ненадання користувачем дозволів, згортання/розгортання додатку, зникнення/поява підключення до мережі, написання хибних параметрів, створити/видалити інформацію, використання різних браузерів тощо.

4.3 Опис тестів

Тестування, описане нижче, здійснюється для веб-додатку системи обліку продукції та вендингових автоматів створеного з використанням мови TypeScript та фреймворку Angular.

Тестування здійснювалося на персональному комп'ютері з операційною системою Windows 10 Pro та екраном LG FullHD та смартфон Huawei P10 lite(Android, Chrome 81.0.4044).

Тестування встановлення описане у таблицях 4.1-4.17. Виконується на персональному комп'ютері з операційною системою Windows 10 Pro та сучасному браузером Chrome 62.3.52. та на смартфоні Huawei P10 lite в браузері Chrome 81.0.4044.138.

Таблиця 4.1 – Завантаження додатку

Мета тесту	Перевірка можливості завантаження додатку.
Початковий стан	В браузер завантажено веб-додатку.
Вхідні дані	https://master.d3btwxooctxbql.amplifyapp.com/

Таблиця 4.2 –Авторизуватися в додатку

Мета тесту	Перевірка можливості авторизуватися в додатку.
Схема проведення тесту	Ввести в поле логін admin@mail.com та "Qwerty123TEST" в поле паролю натиснути кнопку "Login".
Очікуваний результат	Вхід в веб-додаток успішний.
Фактичний результат	Вхід в веб-додаток успішний.

Таблиця 4.3 –Вихід з веб-додатку

Мета тесту	Перевірка можливості виходу з веб-додатку.
Початковий стан	Авторизований та знаходиться на головній сторінці.
Схема проведення тесту	Спробувати вийти з веб-додатку шляхом натискання на кнопку “Logout”.
Очікуваний результат	Успішний вихід з веб-додатку.
Фактичний результат	Успішний вихід з веб-додатку.

Таблиця 4.4 – Додати та заповнити описом новий продукт.

Мета тесту	Перевірка можливість додати на наповнити інформацією новий продукт.
Початковий стан	Авторизований та знаходиться на головній сторінці.
Схема проведення тесту	Розгорнути“Vending Panel” увійти у розділ “Commerce” та відкрити “Products”. Натиснути кнопку “Add new product” та заповнити всі поля актуальною інформацією, натиснути на поле “Enable” та заповнити
	розділ “Inventory”. Натиснути кнопку “Add” В кінці обрати вендингову машину.
Очікуваний результат	Успішне додавання нового продукту та заповнення необхідною інформацією.
Фактичний результат	Успішне додавання нового продукту та заповнення необхідною інформацією.

Таблиця 4.5 – Актуалізація даних по продукту

Мета тесту	Перевірка можливості актуалізації даних щодо продукту.
Початковий стан	Авторизований та знаходиться в розділі “Products”.
Схема проведення тесту	Натиснути на необхідний продукт та редагувати необхідний розділ. Потім натиснути на кнопку “Save”.
Очікуваний результат	Успішне збереження оновленої інформації по продукту.
Фактичний результат	Успішне збереження оновленої інформації по продукту.

Таблиця 4.6 – Пошук необхідного продукту

Мета тесту	Перевірка пошукової системи в розділі “Products”.
Початковий стан	Авторизований та знаходиться в розділі “Products”.
Схема проведення тесту	Натиснути на поле пошуку продукту та ввести слово “HARIBO”.
Очікуваний результат	Знайти коректні 3 позиції по цьому запиту.
Фактичний результат	Знайти коректні 3 позиції по цьому запиту.

Таблиця 4.7 – Видалення продукту

Мета тесту	Перевірка можливості видалення продукту.
Початковий стан	Авторизований та знаходиться в розділі “Products”.
Схема проведення тесту	Знайти необхідний продукт та натиснути на знак “basket”.

Продовження таблиці 4.7.

Очікуваний результат	Успішне видалення продукту зі списку “Products” та видалення зав’язків з вендинговими машинами даної позиції.
Фактичний результат	Успішне видалення продукту зі списку “Products” та видалення зав’язків з вендинговими машинами даної позиції.

Таблиця 4.8 – Додати та заповнити описом новий магазин.

Мета тесту	Перевірка можливість додати на наповнити інформацією новий магазин.
Початковий стан	Авторизований та знаходиться на головній сторінці.
Схема проведення тесту	Розгорнути“Vending Panel” увійти у розділ “Commerce” та відкрити “Stores”. Натиснути кнопку “Add new store/point” та заповнити всі поля актуальною інформацією, натиснути на поле “Enadbled”. В розділі “Machines” обрати машину зі списку та заповнити розділ “Employees”. Натиснути кнопку “Add”.
Очікуваний результат	Успішне додавання нового магазину та заповнення необхідною інформацією.
Фактичний результат	Успішне додавання нового магазину та заповнення необхідною інформацією.

Таблиця 4.9 – Актуалізація даних по продукту

Мета тесту	Перевірка можливості актуалізації даних щодо продукту.
Початковий стан	Авторизований та знаходиться в розділі “Stores”.

Продовження таблиці 4.9.

Схема проведення тесту	Натиснути на необхідний магазин та почати редагувати необхідний розділ. Потім натиснути на кнопку “Save”.
Очікуваний результат	Успішне збереження оновленої інформації по магазину.
Фактичний результат	Успішне збереження оновленої інформації по магазину.

Таблиця 4.10 – Пошук необхідного магазину за назвою

Мета тесту	Перевірка пошукової системи в розділі “Stores”.
Початковий стан	Авторизований та знаходиться в розділі “Stores”.
Схема проведення тесту	Натиснути на поле пошуку продукту та ввести слово “ Lavina ”.
Очікуваний результат	Успішне знаходження потрібного магазину.
Фактичний результат	Успішне знаходження потрібного магазину.

Таблиця 4.11 – Пошук необхідного магазину за адресою

Мета тесту	Перевірка пошукової системи в розділі “Stores”.
Початковий стан	Авторизований та знаходиться в розділі “Stores”.

Продовження таблиці 4.11.

Схема проведення тесту	Натиснути на поле пошуку продукту та ввести слово “ Berkovetska Street, 6D, Kyiv”.
Очікуваний результат	Успішне знаходження торгової точки “Lavina ”.
Фактичний результат	Успішне знаходження торгової точки “Lavina ”.

Таблиця 4.12 – Видалення магазину

Мета тесту	Перевірка можливості видалення магазину.
Початковий стан	Авторизований та знаходиться в розділі “Stores”.
Схема проведення тесту	Знайти необхідний магазин та натиснути на знак “basket”.
Очікуваний результат	Успішне видалення магазину зі списку “Stores” та видалення зв’язку вендингової машина зі списку “Machines”.
Фактичний результат	Успішне видалення магазину зі списку “Stores” та видалення зв’язку вендингової машина зі списку “Machines”.

Таблиця 4.13 – Додати та заповнити описом нову вендингову машину

Мета тесту	Перевірка можливість додати на наповнити інформацією нову вендингову машину.
Початковий стан	Авторизований та знаходиться на головній сторінці.
Схема проведення тесту	Розгорнути “Vending Panel” увійти у розділ “Commerce” та відкрити “Machines”. Натиснути кнопку “Add new machines” та заповнити всі поля актуальною інформацією, натиснути на поле “Enadbled”. Натиснути кнопку “Add”. В розділі “Inventory” обрати необхідні продукти та заповнити розділ.
Очікуваний результат	Успішне додавання нової вендингової машини та заповнення необхідною інформацією.

Продовження таблиці 4.13.

Фактичний результат	Успішне додавання нової вендингової машини та заповнення необхідною інформацією.

Таблиця 4.14 – Актуалізація даних по вендинговій машині

Мета тесту	Перевірка можливості актуалізації даних щодо вендингової машини.
Початковий стан	Авторизований та знаходиться в розділі “Machines”.
Схема проведення тесту	Натиснути на необхідну машину та почати редагувати необхідний розділ. Потім натиснути на кнопку “Save”.
Очікуваний результат	Успішне збереження оновленої інформації по вендинговій машині.
Фактичний результат	Успішне збереження оновленої інформації по вендинговій машині.

Таблиця 4.15 – Пошук необхідного вендингового автомата за назвою машини

Мета тесту	Перевірка пошукової системи в розділі “Machines”.
Початковий стан	Авторизований та знаходиться в розділі “Machines”.
Схема проведення тесту	Натиснути на поле пошуку продукту та ввести слово “ Dixie Narco Model 5591-Bevmax”.
Очікуваний результат	Знайти 10 позиції по цьому запиту.
Фактичний результат	Знайти 10 позиції по цьому запиту.

Таблиця 4.16 – Пошук необхідного вендингового автомата за назвою магазина

Мета тесту	Перевірка пошукової системи в розділі “Machines”.
Початковий стан	Авторизований та знаходиться в розділі “Machines”.
Схема проведення тесту	Натиснути на поле пошуку продукту та ввести слово “ HOLLYWOOD ”.
Очікуваний результат	Знайти 4 позиції по цьому запиту.
Фактичний результат	Знайти 4 позиції по цьому запиту.

Таблиця 4.17 – Видалення вендингового автомата

Мета тесту	Перевірка можливості видалення вендингового автомата.
Початковий стан	Авторизований та знаходиться в розділі “Machines”.
Схема проведення тесту	Знайти необхідний магазин та натиснути на знак “basket”.
Очікуваний результат	При успішному видаленню вендингового автомата, розриваються зв’язки з продуктами, які містились в автоматі та з магазинами в яких він обслуговувався.
Фактичний результат	При успішному видаленню вендингового автомата, розриваються зв’язки з продуктами, які містились в автоматі та з магазинами в яких він обслуговувався.

Тестування сумісності, функціональне тестування та тестування зручності було виконано на наступних пристроях:

– huawei p10 lite;

– macbook 13;

– lenovo T470.

Результати тестів не відрізняються від очікуваних результатів, описаних у тестах, які були наведенні вище.

При тестуванні додатку на низькому рівень заряду акумулятора пристрою не має впливати на правильність роботи веб-додатку. Результати тестів при низькому рівні заряду не відрізняються від очікуваних результатів.

Тестування переривання. Виконання всіх попередніх Тестування зручності використання тестів на ПК з ОС Windows 10 Pro з:

– зміна екрану;

– зміною розміру застосунку та розгортанням застосунку;

– блокування/розблокування пристрою.

перериванням за допомогою консолі або іншим застосунком.

Будь-які переривання не мають впливати на правильність роботи веб-додатку. Результати тестів з додавання переривань не відрізняються від очікуваних результатів.

4.4 Висновки до розділу

Не зважаючи на те що, при тестуванні веб-додатку, потрібно враховувати деякі особливості: різні версії ОС, різне підключення до мережі Інтернет, різні розміри екрану, раптові роз’єднання з інтернет зв’язком, апаратні відмінності тощо. Проте тестування додатку – є не від’ємною частиною розробки. Адже, це допомагає розробникам: швидке виправлення помилок і несправностей, швидка розробка проєкту, що призводить до зменшення витрат та рефакторинг коду. Тільки при тестуванні додатку, можна впевнитись в зручності використання та перевірити очікувану функціональну поведінку, перш ніж презентувати продукт. Вище наведені тести, які перевіряють зручність використання, сумісності з різними системами, переривання з мережею та основні функції веб-додатку.

5 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розгортання програмного забезпечення

Для розгортання даного веб-додатку потрібні такі параметри компьютера:

– oc: Windows, Linuxe, MacOS, Andorid

– cli: bash (для ОС Linuxe), cmd або PowerShell 6, 7 (для ОС Windows), Termux (для ОС Andorid)

– browser: Chrome 81, Opera 68, Firefox 76, Chrome Mobile 81

– програмне середовище node.js 10+, Git

Необхідно мати попереднь налаштований аккаунт AWS для подальшої розгортання проєту.

Пропускна лінія Інтернету повина бути від 300кб та вище, щоб завантажити усі необхідні бібліотеки для попальшого компілювання веб-додатку. На рисунку 5.1 зображено клонування проету с системи конролю версій Git.

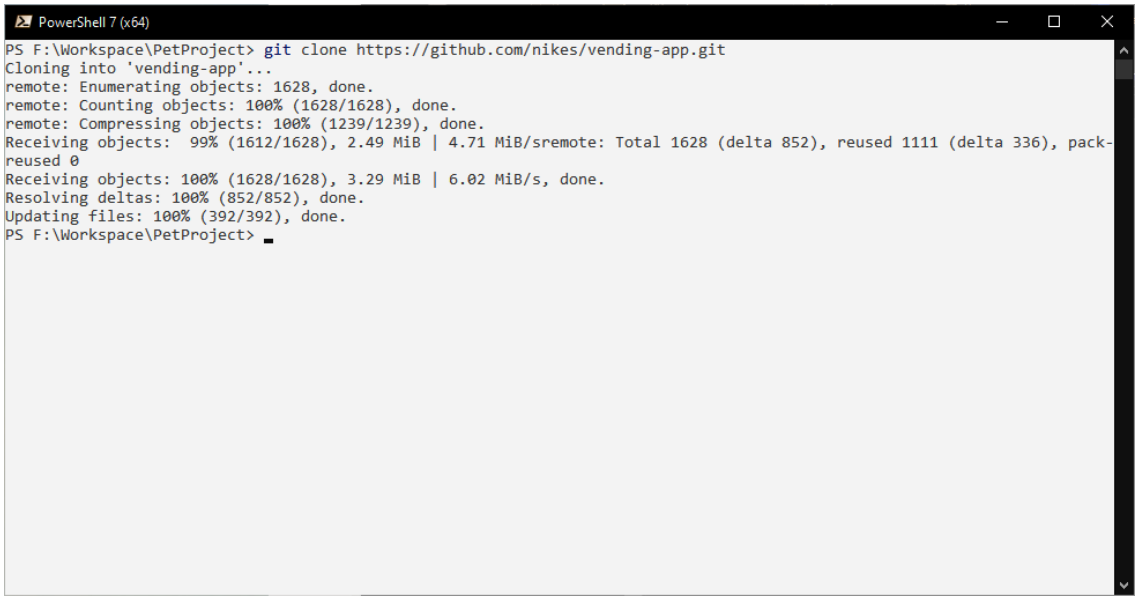

Рисунок 5.1 – Клонування проєкта з системи контролю версій

Після клонування проєкта з GitHub, переходимо до клонованого проекту та виконуємо насталювання усіх необхідних пакетів за допомогою пакетного менеджера NPM, який входить у програмне середовище Node.JS.

На рисунку 5.2 зображено команду для встановлення усіх необхідних залежностей.

Після встановлення залежностей необхідно розгорнути проект за допомогою консольних команд, які зображені на рисунку 5.3 – 5.4. Після введення команди почнеться автоматичне розгортання проекту до AWS Amplify середовища.

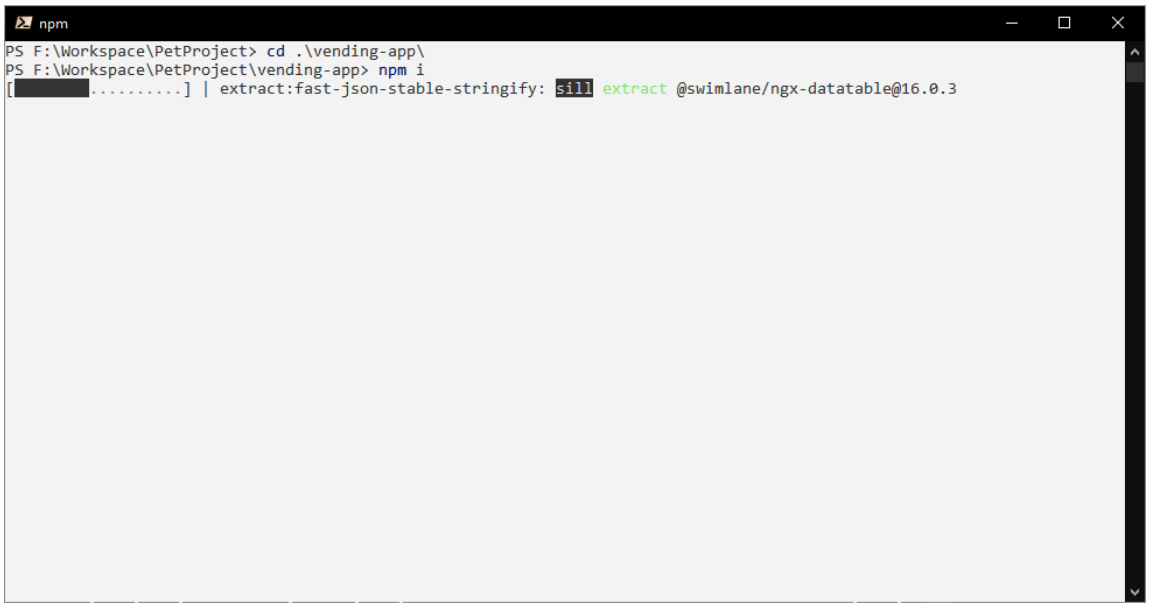

Рисунок 5.2 – Команда для налаштування усіх необхідних залежностей

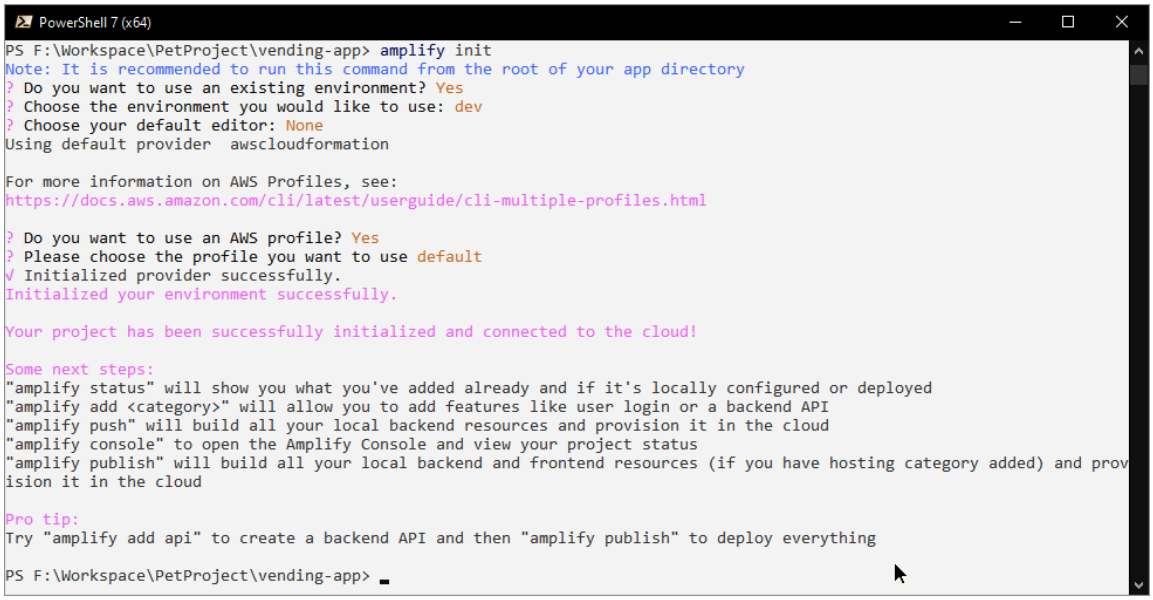

Рисунок 5.3 – Ініціалізація веб-додатку у хмарі

Після розгортання веб-додатку, перейдемо до хмари, та перейдемо до самого адресу веб-додатку.

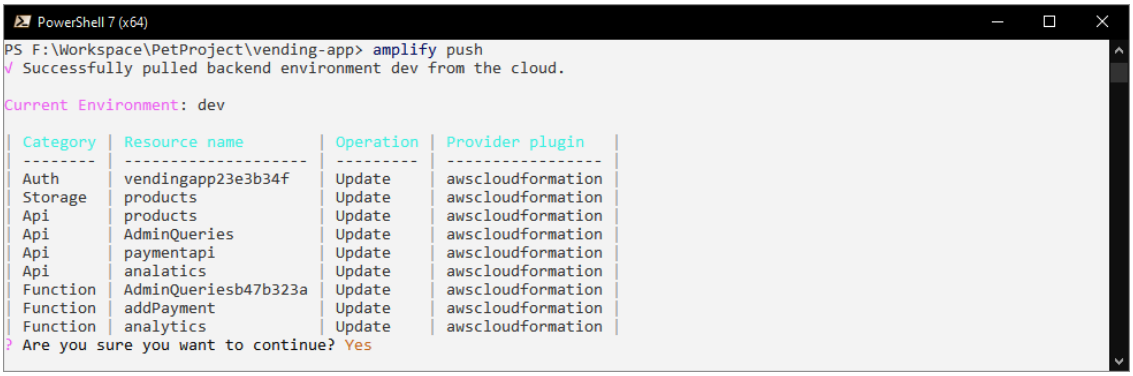

Рисунок 5.4 – Публікація проекту у сервісі AWS Amplify

На рисунку 5.5 зображена консоль AWS Amplify та список веб-додатків, а на рисунку 5.6 зображені панель налаштувань веб-додатку.

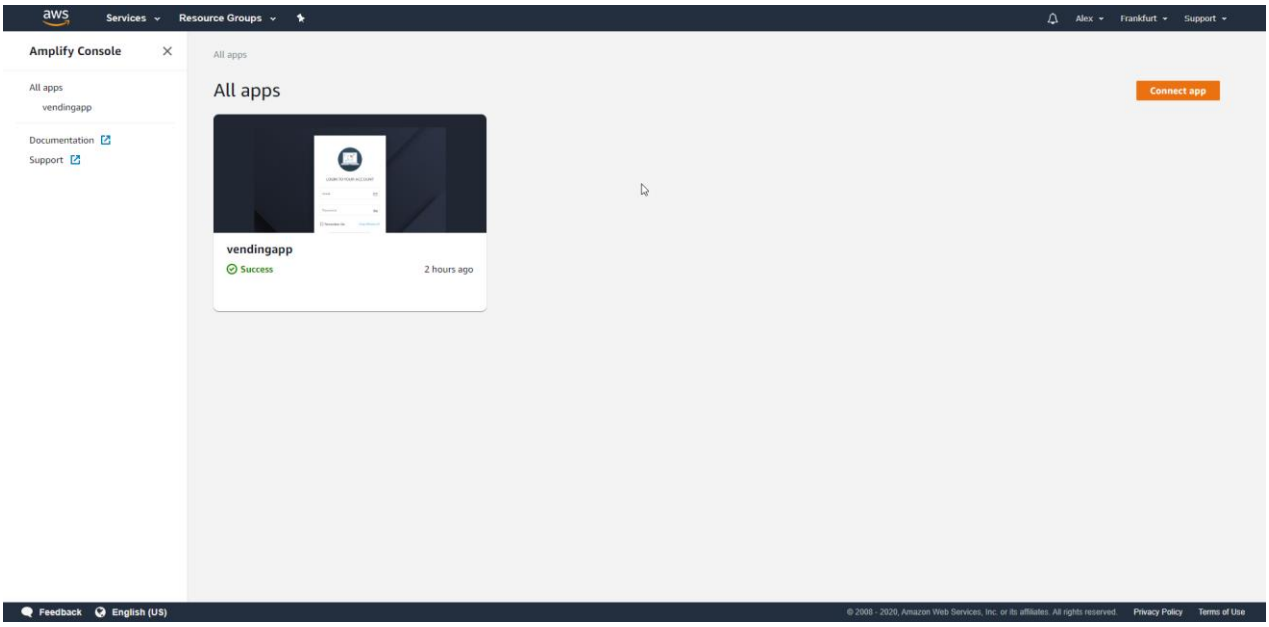

Рисунок 5.5 – Консоль AWS Amplify, відображення списку додатків

Після переходу до веб додатка відображається форма авторизації, яка відображена на рисунку 5.7.

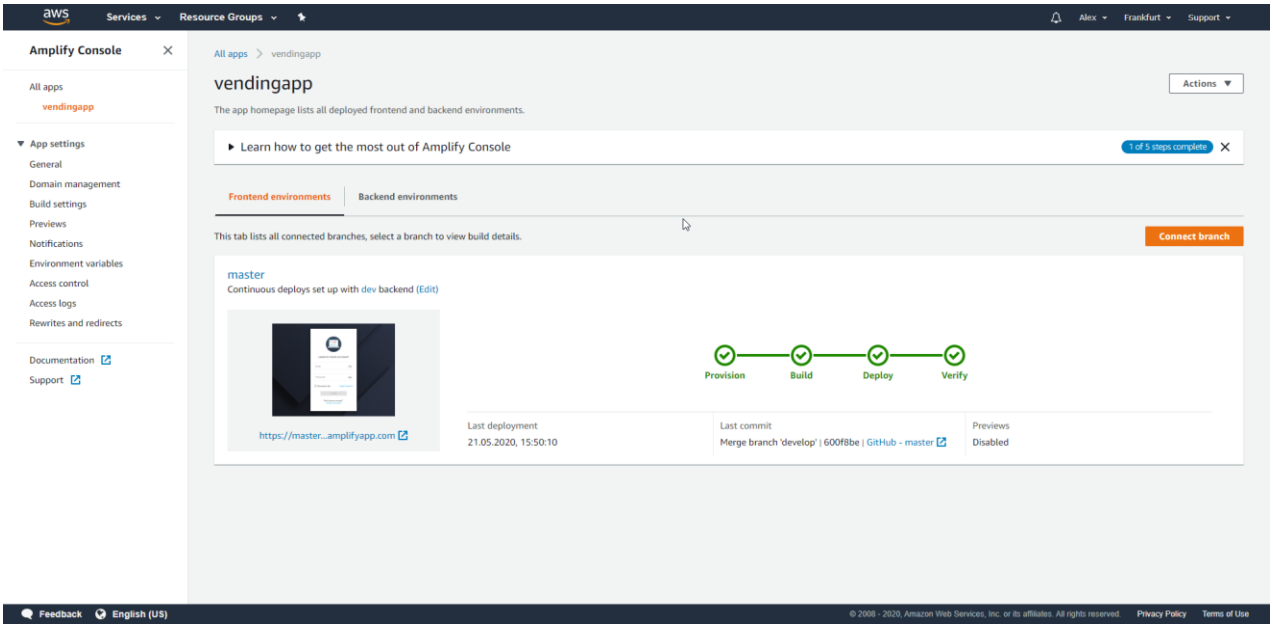

Рисунок 5.6 – Налаштування веб-додатку у хмарі

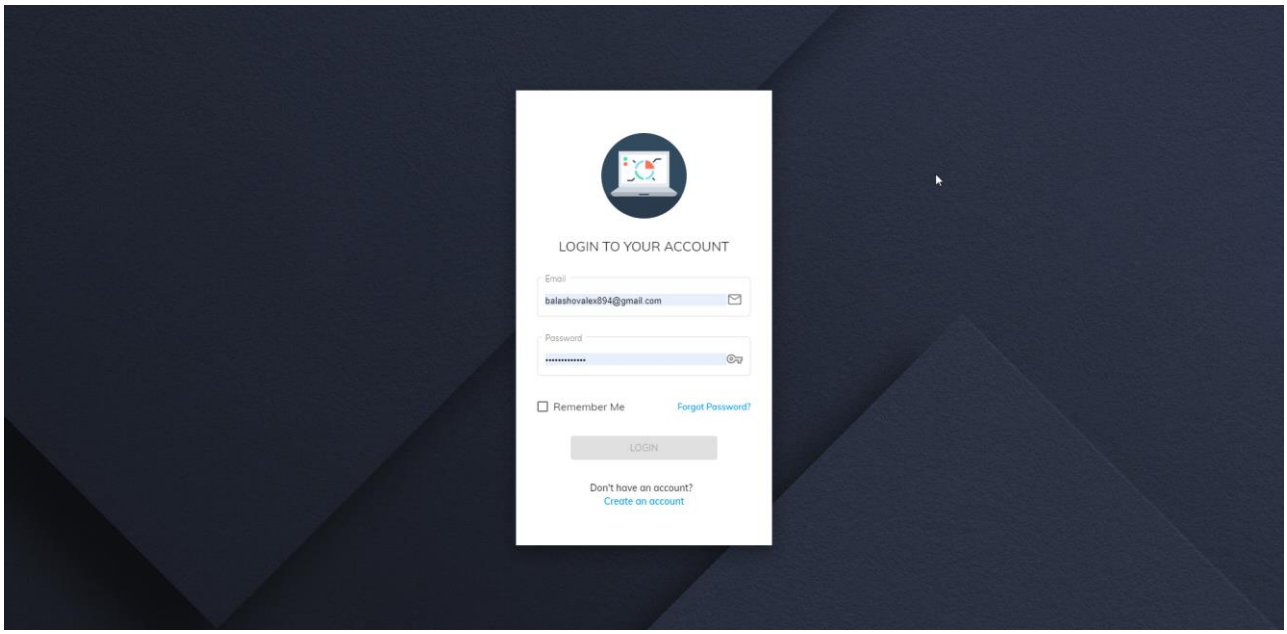

Рисунок 5.7 – Сторінка авторизації до веб-додатку

Щоб продовжити роботу з системою, необхідно зареєструватися та за допомогою AWS Cognito необхідно додати до цього користувача політику адміністратора системи. Приклад реєстрації працівника зображено на рисунку 5.8.

5.2 Робота з програмним забезпеченням

Для входу до системи, необхідно зареєструватися та заповнити усі необхідні поля та натиснути на кнопку “CREATE AN ACCOUNT”. Приклад реєстрації працівника зображено на рисунку 5.8.

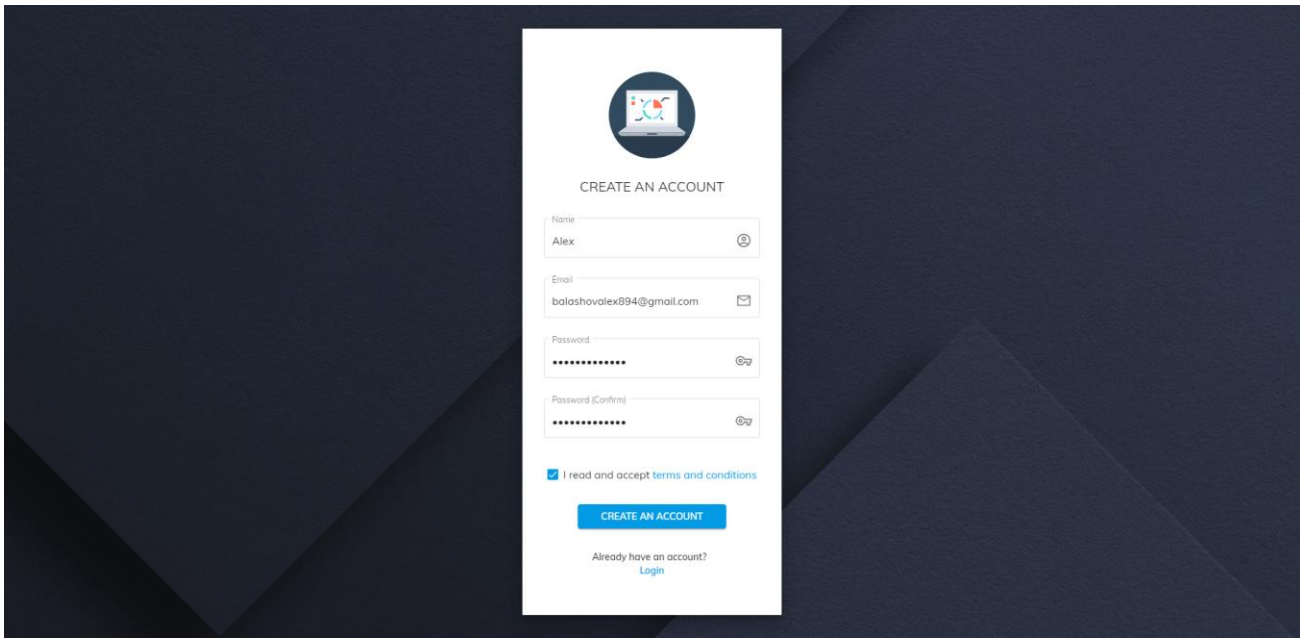

Рисунок 5.8 – Сторінка реєстрації працівника

Після реєстрації переходимо на сторінку авторизації, приклад якої зображено на рисунку 5.7.

Після проходження авторизації, система автоматично переходить до сторінки статистики операцій. Приклад зображено на рисунку 5.9.

У навігації веб-додатком, розгорнути “Commerce” та перейти на сторінку “Stores”. Натиснувши на кнопку “ADD NEW STORE/POINT” буде створено нову точку або магазин, при використані кнопки “DELETE” буде видалено цей запис. Приклад сторінки точок, додавання та редагування наведенні на рисунках 5.10 - 5.11.

Усі інші сторінки “Machines”, “Products” мають аналогічний функціонал та інтерфейс.

Для приєднання працівника до точки або магазину необхідно перейти до пункту навігації “User Pool” та відкрити сторінку “Users”. Після необхідно

знайти необхідного працівника та додати необхідний магазин до списку працівника.

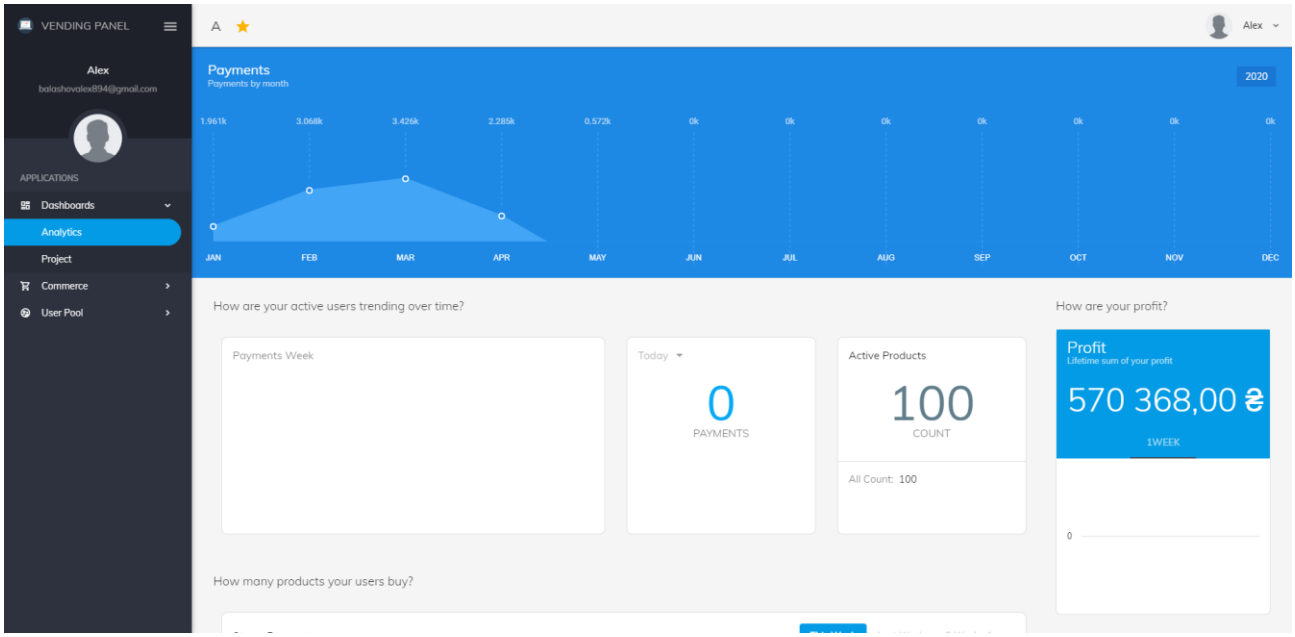

Рисунок 5.9 – Сторінка статистики

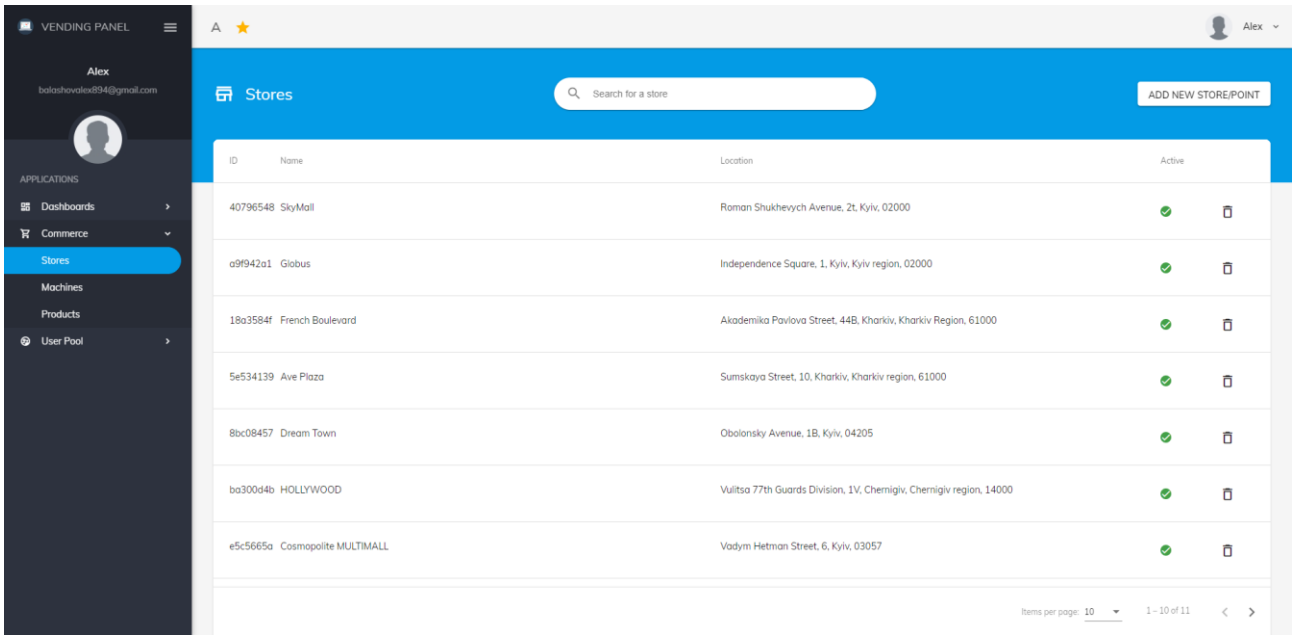

Рисунок 5.10 – Сторінка списку магазинів

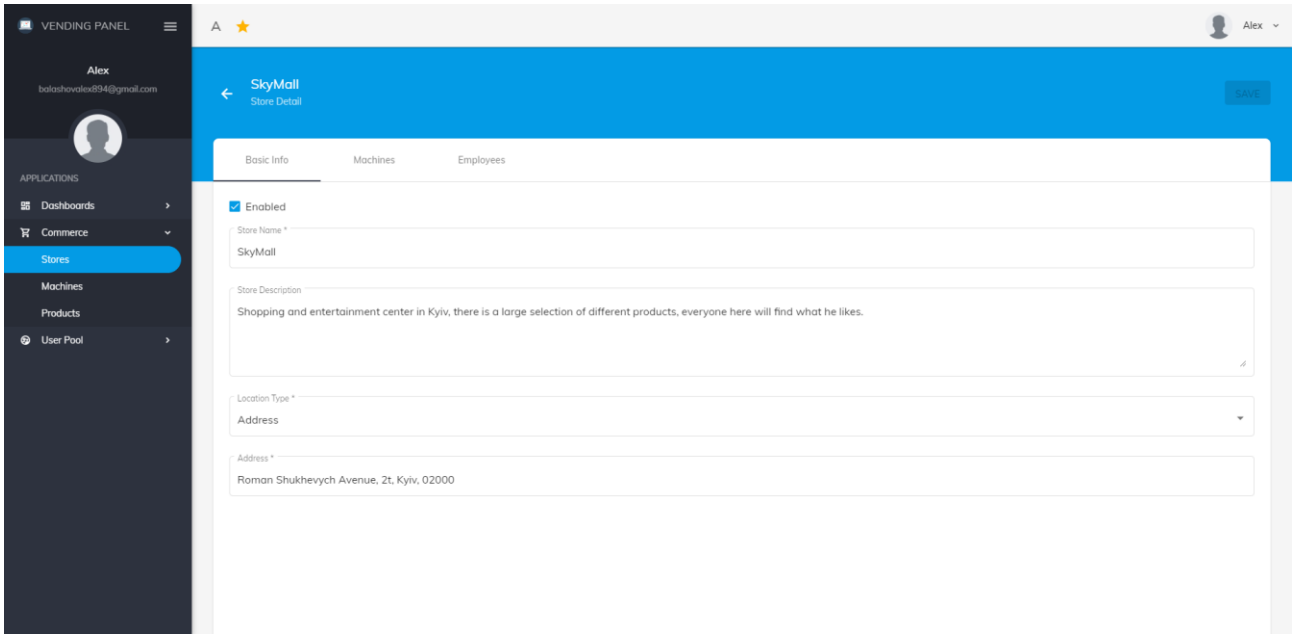

Рисунок 5.11 – Сторінка редагування магазинів або точок

5.3 Супровід програмного забезпечення

Після публікації у GitHub для контролю помилок використовується онлайн сервіс AWS Console, який веде усі журнали як веб-додатку так і серверної частини. При виявленні помилок на етапі розгортання або використання веб-додатку, необхідно створити у проєкті на GitHub issue з детальним описом виявленої помилки для подальшого вирішення даної ситуації.

5.4 Висновки до розділу

У даному розділі було описано етапи розгортання даного веб-застосунку, наведено інформацію стосовно використання системи та вказані засоби супроводу.

ВИСНОВКИ

У ході виконання дипломного проєкту було розроблено автоматизовану систему контроля операцій вендингових точок.

Першим етапом розробки цього проєкту був проведений аналіз предметної області та порівняльна характеристика існуючих рішень. Було проведено аналіз та порівняння існуючих технічних рішень, та програмного забезпечення у сфері сучасних веб фреймворків та автоматизованих систем. Також розглянуті хмарні технології та детально проаналізовані відомі хмарні рішення та їх сервісні моделі.

На основі проведеного аналізу було виявлені переваги та недоліки та стало можливим обрати найкращі рішення для реалізації автоматизованої системи контролю операцій вендингових точок, а так же сформовані вимоги.

Для реалізації веб-додатку було обрано фреймворк Angular, який підтримує мову суворої типізації TypeScript, а для реалізації серверної частини автоматизованої системи, хмарний сервіс Amazon Web Services.

Описано засоби розробки, архітектура програмного забезпечення, інструкція користувача та розгортання автоматизованої системи.

Спроектований і розроблений веб-додаток для будь якої операційної системи та будь-яким сучасним браузером для значного полегшення використання автоматизованої системи на підприємстві.

Після завершення основного етапу розробки веб-додатку було проведено тестування автоматизованої системи.

На основі проведеного аналізу, моделювання та розробки веб-додатку, було розроблено проєктну документацію, діаграми архітектури системи, методики тестування, інструкцію користувача та опис розгортання системи.

Таким чином усі поставленні завдання даного дипломного проєкту були виконанні і була досягнута поставлена мета – створення простої автоматизованої системи контролю операцій вендингових точок за допомогою сучасних хмарних рішень.

ПЕРЕЛІК ПОСИЛАНЬ

1. What is a vending machine? [Електронний ресурс] – Режим доступу: http://web.mit.edu/2.744/studentSubmissions/humanUseAnalysis/keval/whatisvm.html

2. Початок роботи – React [Електронний ресурс] – Режим доступу: https://uk.reactjs.org/docs/getting-started.html

3. Руководство по React [Електронний ресурс] – Режим доступу: https://metanit.com/web/react/

4. Руководство по началу работы для разработчиков Azure [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/ru-ru/azure/guides/developer/azure-developer-guide

5. Angular - Introduction to the Angular Docs [Електронний ресурс] – Режим доступу: https://angular.io/docs

6. Руководство по Angular 9 [Електронний ресурс] – Режим доступу: https://metanit.com/web/angular2

7. Введение — Vue.js [Електронний ресурс] – Режим доступу: https://ru.vuejs.org/v2/guide/index.html

8. Руководство по Vue.js [Електронний ресурс] – Режим доступу: https://metanit.com/web/vuejs

9. Огляд системи 1С:Підприємство 8 [Електронний ресурс] – Режим доступу: http://1c.ua/ua/v8/index.php

10. Управління торговим підприємством [Електронний ресурс] – Режим доступу: http://1c.ua/ua/v8/RegionalSolutions_UA_UTP.php

11. SAP что это такое? Описание модулей системы САП [Електронний ресурс] – Режим доступу: http://asapcg.com/press-center/articles/chto-takoe-sap-sistemy/

12. SAP опис архітектури [Електронний ресурс] – Режим доступу: http://asapcg.com/press-center/articleserp-sistemy-sap/

13. Что такое Dynamics 365 [Електронний ресурс] – Режим доступу: https://dynamics.microsoft.com/ru-ru/what-is-dynamics365/

14. Об AWS [Електронний ресурс] – Режим доступу: https://aws.amazon.com/ru/about-aws/

15. Почему Windows Azure стала просто Azure? Введение в платформу для пользователей Linux, Open Source, Oracle DB, Android, iOS и других инструментов [Електронний ресурс] – Режим доступу: https://habr.com/ru/company/microsoft/blog/219925/

16. Архитектура REST [Електронний ресурс] – Режим доступу: https://habr.com/ru/post/38730/

17. CRAP and CRUD: From Database to Datacloud [Електронний ресурс] – Режим доступу: https://blog.dellemc.com/en-us/crap-and-crud-from-database-to-datacloud/

18. GraphQL | A query language for your API [Електронний ресурс] – Режим доступу: https://graphql.org/

19. AWS Lambda – Бессерверные вычисления – Amazon Web Services [Електронний ресурс] – Режим доступу: https://aws.amazon.com/ru/lambda/

20. Robert C. Martin. Clean Code: A Handbook of Agile Software Craftsmanship. — Pearson Education, 2008. — 157 с.

21. Angular – Introduction to components and templates [Електронний ресурс] - Режим доступу: https://angular.io/guide/architecture-components

22. Angular – Introduction to services and dependency injection [Електронний ресурс] – Режим доступу: https://angular.io/guide/architecture-services

23. AWS Amplify – Режим доступу: https://aws.amazon.com/ru/amplify/

24. Amazon Cognito – Режим доступу: https://aws.amazon.com/ru/cognito/

25. React или Angular или Vue.js — что выбрать? / Хабр – Режим доступу: https://habr.com/ru/post/476312/

26. Angular - Introduction to Angular concepts [Електронний ресурс] – Режим доступу: https://angular.io/guide/architecture#whats-next

27. angular, react, vue, angularjs, next.js - Анализ - Google Trends [Електронний ресурс] – Режим доступу: https://trends.google.ru/trends/explore?date=today%205-y&q=angular,react,vue,angularjs,next.js

28. Google Cloud Platform, Microsoft Azure, Oracle Cloud, Amazon Web Services - Анализ - Google Trends [Електронний ресурс] – Режим доступу: https://trends.google.ru/trends/explore?date=today%205-y&q=%2Fm%2F0105pbj4,%2Fm%2F04y7lrx,%2Fg%2F11h9q1kdd2,%2Fm%2F05nrgx

ДОДАТОК А

Тексти основних програмних модулів

А.1 Програмний код головного компоненту

```
import { Component, Inject, OnDestroy } from '@angular/core';
import { DOCUMENT } from '@angular/common';
import { Platform } from '@angular/cdk/platform';
import { Router } from '@angular/router';

import { AmplifyService } from 'aws-amplify-angular';
import { Subject } from 'rxjs';
import { takeUntil } from 'rxjs/operators';

import { FuseConfigService } from '@fuse/services/config.service';
import { FuseNavigationService } from '@fuse/components/navigation/navigation.service';
import { FuseSidebarService } from '@fuse/components/sidebar/sidebar.service';
import { FuseSplashScreenService } from '@fuse/services/splash-screen.service';

import { navigation } from 'app/navigation/navigation';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent implements OnDestroy {
  private _destroyed$: Subject<boolean> = new Subject<boolean>();

  fuseConfig: any;
  navigation: any;

  constructor(private _router: Router,
              private _amplifyService: AmplifyService,
              @Inject(DOCUMENT) private document: any,
              private _fuseConfigService: FuseConfigService,
              private _fuseNavigationService: FuseNavigationService,
              private _fuseSidebarService: FuseSidebarService,
              private _fuseSplashScreenService: FuseSplashScreenService,
              private _platform: Platform) {
    this._amplifyService
      .authStateChange$
      .pipe(takeUntil(this._destroyed$))
      .subscribe(authState => {
        console.log(`[AuthState]`, authState);
        switch (authState.state) {
          case 'confirmSignUp':
            const user = authState.user.username;
            return this._router.navigate(['pages', 'auth', 'mail-confirm'], {
              queryParams: {
                email: user.attributes.email
              }
            });
          case 'signIn':
            const params: any = {};

            if (authState.user && authState.user.username) {
              params.email = authState.user.username;
            }

            return this._router.navigate(['pages', 'auth', 'login'], {
              queryParams: params
            });
          case 'signedIn':
            return this._router.navigate(['apps', 'dashboards', 'analytics']);
          case 'signedOut':
```

```
          return this._router.navigate(['pages', 'auth', 'login']);
        default:
          return console.error(`[STATE] ${authState.state}`, authState);
      }
    });

  // Get default navigation
  this.navigation = navigation;

  // Register the navigation to the service
  this._fuseNavigationService.register('main', this.navigation);

  // Set the main navigation as our current navigation
  this._fuseNavigationService.setCurrentNavigation('main');

  /**
   * ----------------------------------------------------------------------------------------------------
   * ngxTranslate Fix End
   * ----------------------------------------------------------------------------------------------------
   */

  // Add is-mobile class to the body if the platform is mobile
  if (this._platform.ANDROID || this._platform.IOS) {
    this.document.body.classList.add('is-mobile');
  }
}

/**
 * On init
 */
ngOnInit(): void {
  // Subscribe to config changes
  this._fuseConfigService.config
    .pipe(takeUntil(this._destroyed$))
    .subscribe((config) => {

      this.fuseConfig = config;
      // Boxed
      if (this.fuseConfig.layout.width === 'boxed') {
        this.document.body.classList.add('boxed');
      } else {
        this.document.body.classList.remove('boxed');
      }

      // Color theme - Use normal for loop for IE11 compatibility
      for (let i = 0; i < this.document.body.classList.length; i++) {
        const className = this.document.body.classList[i];

        if (className.startsWith('theme-')) {
          this.document.body.classList.remove(className);
        }
      }

      this.document.body.classList.add(this.fuseConfig.colorTheme);
    });
}

/**
 * On destroy
 */
ngOnDestroy(): void {
  // Unsubscribe from all subscriptions
  this._destroyed$.next(true);
  this._destroyed$.complete();
}

// -----------------------------------------------------------------------------------------------------
// @ Public methods
// -----------------------------------------------------------------------------------------------------

/**
```

```
   * Toggle sidebar open
   *
   * @param key
   */
  toggleSidebarOpen(key): void {
    this._fuseSidebarService.getSidebar(key).toggleOpen();
  }
}
```

А.2 Програмний код компоненту аналітики

```
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
import * as shape from 'd3-shape';

import { fuseAnimations } from '@fuse/animations';

import { AnalyticsDashboardService } from 'app/main/apps/dashboards/analytics/analytics.service';

@Component({
  selector: 'analytics-dashboard',
  templateUrl: './analytics.component.html',
  styleUrls: ['./analytics.component.scss'],
  encapsulation: ViewEncapsulation.None,
  animations: fuseAnimations
})
export class AnalyticsDashboardComponent implements OnInit {
  analyticWidgets: any = {
    widget1: {
      chartType: 'line',
      datasets: {
        '2020': [
          {
            label: 'Payments',
            data: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
            fill: 'start'
          }
        ]
      },
      labels: ['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL', 'AUG', 'SEP', 'OCT', 'NOV', 'DEC'],
      colors: [
        {
          borderColor: '#42a5f5',
          backgroundColor: '#42a5f5',
          pointBackgroundColor: '#1e88e5',
          pointHoverBackgroundColor: '#1e88e5',
          pointBorderColor: '#ffffff',
          pointHoverBorderColor: '#ffffff'
        }
      ],
      options: {
        spanGaps: false,
        legend: {
          display: false
        },
        maintainAspectRatio: false,
        layout: {
          padding: {
            top: 32,
            left: 32,
            right: 32
          }
        },
        elements: {
          point: {
            radius: 4,
            borderWidth: 2,
            hoverRadius: 4,
            hoverBorderWidth: 2
          },
          line: {
            tension: 0
```

```
      }
    },
    scales: {
      xAxes: [
        {
          gridLines: {
            display: false,
            drawBorder: false,
            tickMarkLength: 18
          },
          ticks: {
            fontColor: '#ffffff'
          }
        }
      ],
      yAxes: [
        {
          display: false,
          ticks: {
            min: 1.5,
            max: 5,
            stepSize: 0.5
          }
        }
      ]
    },
    plugins: {
      filler: {
        propagate: false
      },
      xLabelsOnTop: {
        active: true
      }
    }
  }
},
widget4: {
  chartType: 'bar',
  datasets: [
    {
      label: 'Payments',
      data: [0, 0, 0, 0, 0, 0, 0],
    }
  ],
  labels: ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'],
  colors: [
    {
      borderColor: '#f44336',
      backgroundColor: '#f44336'
    }
  ],
  options: {
    spanGaps: false,
    legend: {
      display: false
    },
    maintainAspectRatio: false,
    layout: {
      padding: {
        top: 24,
        left: 16,
        right: 16,
        bottom: 16
      }
    },
    scales: {
      xAxes: [
        {
          display: false
        }
      ],
```

```
        yAxes: [
          {
            display: false
          }
        ]
      }
    }
  },
  widget8: {
    scheme: {
      domain: ['#5c84f1']
    },
    today: '0',
    data: [
      {
        name: 'Profit',
        series: []
      }
    ]
  }
};
projectWidgets: any = {
  widget1: {
    ranges: {
      'DY': 'Yesterday',
      'DT': 'Today',
      'DTM': 'Tomorrow'
    },
    currentRange: 'DT',
    data: {
      label: 'PAYMENTS',
      count: {
        'DY': 0,
        'DT': 0,
        'DTM': 0
      }
    }
  },
  widget4: {
    title: 'Active Products',
    data: {
      label: 'COUNT',
      count: 0,
      extra: {
        label: 'All Count',
        count: 0
      }
    }
  },
  widget5: {
    title: 'Stores Payments',
    ranges: {
      'TW': 'This Week',
      'LW': 'Last Week',
      '2W': '2 Weeks Ago'
    },
    mainChart: {
      '2W': [],
      'LW': [],
      'TW': []
    }
  },
  widget6: {
    title: 'Purchased Products',
    ranges: {
      'TW': 'This Week',
      'LW': 'Last Week',
      '2W': '2 Weeks Ago'
    },
    mainChart: {
      'TW': [],
```

```
      'LW': [],
      '2W': []
    }
  },
};

widget1SelectedYear = '2020';

widget5: any = {};
widget6: any = {};
widget7: any = {};
widget8: any = {};
widget9: any = {};


/**
 * Constructor
 *
 * @param {AnalyticsDashboardService} _analyticsDashboardService
 */
constructor(
  private _analyticsDashboardService: AnalyticsDashboardService
) {
  // Register the custom chart.js plugin
  this._registerCustomChartJSPlugin();

  /**
   * Widget 5
   */
  this.widget5 = {
    currentRange: 'TW',
    xAxis: true,
    yAxis: true,
    gradient: false,
    legend: false,
    showXAxisLabel: false,
    xAxisLabel: 'Days',
    showYAxisLabel: false,
    yAxisLabel: 'Isues',
    scheme: {
      domain: ['#42BFF7', '#C6ECFD', '#C7B42C', '#AAAAAA']
    },
    onSelect: (ev) => {
      console.log(ev);
    },
    supporting: {
      currentRange: '',
      xAxis: false,
      yAxis: false,
      gradient: false,
      legend: false,
      showXAxisLabel: false,
      xAxisLabel: 'Days',
      showYAxisLabel: false,
      yAxisLabel: 'Isues',
      scheme: {
        domain: ['#42BFF7', '#C6ECFD', '#C7B42C', '#AAAAAA']
      },
      curve: shape.curveBasis
    }
  };

  /**
   * Widget 6
   */
  this.widget6 = {
    currentRange: 'TW',
    legend: false,
    explodeSlices: false,
    labels: true,
    doughnut: true,
```

```
    gradient: false,
    scheme: {
      domain: ['#f44336', '#9c27b0', '#03a9f4', '#e91e63']
    },
    onSelect: (ev) => {
      console.log(ev);
    }
  };

  /**
   * Widget 7
   */
  this.widget7 = {
    currentRange: 'T'
  };

  /**
   * Widget 8
   */
  this.widget8 = {
    legend: false,
    explodeSlices: false,
    labels: true,
    doughnut: false,
    gradient: false,
    scheme: {
      domain: ['#f44336', '#9c27b0', '#03a9f4', '#e91e63', '#ffc107']
    },
    onSelect: (ev) => {
      console.log(ev);
    }
  };

  /**
   * Widget 9
   */
  this.widget9 = {
    currentRange: 'TW',
    xAxis: false,
    yAxis: false,
    gradient: false,
    legend: false,
    showXAxisLabel: false,
    xAxisLabel: 'Days',
    showYAxisLabel: false,
    yAxisLabel: 'Isues',
    scheme: {
      domain: ['#42BFF7', '#C6ECFD', '#C7B42C', '#AAAAAA']
    },
    curve: shape.curveBasis
  };
}

// -----------------------------------------------------------------------------------------------
// @ Lifecycle hooks
// -----------------------------------------------------------------------------------------------

/**
 * On init
 */
ngOnInit(): void {
  const year = new Date().getFullYear();
  // Get the widgets from the service
  const analytics = this._analyticsDashboardService.analytics;
  const activeProduct = this._analyticsDashboardService.activeProduct;

  if (analytics) {
    this.analyticWidgets.widget1.datasets[year][0].data = analytics.payments_by_months;
    this.analyticWidgets.widget4.datasets[0].data = analytics.payments_week;
    this.analyticWidgets.widget8.today = analytics.total.cost;
    this.analyticWidgets.widget8.data[0].series = analytics.sales_week;
```

```
    this.projectWidgets.widget1.data.count['DT'] = analytics.today.payments_today;
    this.projectWidgets.widget5.mainChart['TW'] = analytics.store_payments_week;
    this.projectWidgets.widget6.mainChart['TW'] = analytics.products_week;
   }

   if (activeProduct) {
    this.projectWidgets.widget4.data.count = activeProduct.active_count;
    this.projectWidgets.widget4.data.extra.count = activeProduct.all_count;
   }
  }

  // ---------------------------------------------------------------------------------------------------
  // @ Private methods
  // ---------------------------------------------------------------------------------------------------

  /**
   * Register a custom plugin
   */
  private _registerCustomChartJSPlugin(): void {
   (window as any).Chart.plugins.register({
    afterDatasetsDraw: function (chart, easing): any {
     // Only activate the plugin if it's made available
     // in the options
     if (
      !chart.options.plugins.xLabelsOnTop ||
      (chart.options.plugins.xLabelsOnTop && chart.options.plugins.xLabelsOnTop.active === false)
     ) {
      return;
     }

     // To only draw at the end of animation, check for easing === 1
     const ctx = chart.ctx;

     chart.data.datasets.forEach(function (dataset, i): any {
      const meta = chart.getDatasetMeta(i);
      if (!meta.hidden) {
       meta.data.forEach(function (element, index): any {

        // Draw the text in black, with the specified font
        ctx.fillStyle = 'rgba(255, 255, 255, 0.7)';
        const fontSize = 13;
        const fontStyle = 'normal';
        const fontFamily = 'Roboto, Helvetica Neue, Arial';
        ctx.font = (window as any).Chart.helpers.fontString(fontSize, fontStyle, fontFamily);

        // Just naively convert to string for now
        const dataString = dataset.data[index].toString() + 'k';

        // Make sure alignment settings are correct
        ctx.textAlign = 'center';
        ctx.textBaseline = 'middle';
        const padding = 15;
        const startY = 24;
        const position = element.tooltipPosition();
        ctx.fillText(dataString, position.x, startY);

        ctx.save();

        ctx.beginPath();
        ctx.setLineDash([5, 3]);
        ctx.moveTo(position.x, startY + padding);
        ctx.lineTo(position.x, position.y - padding);
        ctx.strokeStyle = 'rgba(255,255,255,0.12)';
        ctx.stroke();

        ctx.restore();
       });
      }
     });
    }
```

```
    });
  }
}
```

А.3 Програмний код провайдер сервісу даних для аналітики

```
import { Injectable } from '@angular/core';
import { ActivatedRouteSnapshot, Resolve, RouterStateSnapshot } from '@angular/router';
import { Observable } from 'rxjs';
import { API } from 'aws-amplify';

import { APIService } from '../../../../API.service';

@Injectable()
export class AnalyticsDashboardService implements Resolve<any> {
  analytics: any;
  activeProduct: any;

  /**
   * Constructor
   *
   * @param {APIService} _api
   */
  constructor(
    private _api: APIService
  ) {
  }

  /**
   * Resolver
   *
   * @param {ActivatedRouteSnapshot} route
   * @param {RouterStateSnapshot} state
   * @returns {Observable<any> | Promise<any> | any}
   */
  resolve(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable<any> | Promise<any> | any {
    return new Promise((resolve, reject) => {

      Promise.all([
        this.getAnalytics(),
        this.getActiveProducts()
      ]).then(
        () => {
          resolve();
        },
        reject
      );
    });
  }

  getActiveProducts(): Promise<void> {
    let allCount = 0;
    let disabledCount = 0;
    return this._api.ListMachineProductss(null, 100)
      .then(response => {
        allCount = response.items.length; // all count
        return this._api.ListMachineProductss({quantity: {eq: 0}}, 100);
      })
      .then(response => {
        disabledCount = response.items.length;

        this.activeProduct = {
          active_count: allCount - disabledCount,
          all_count: allCount
        };
      });
  }

  getAnalytics(): Promise<void> {
```

```
    const year = new Date().getFullYear();
    const from = new Date(year, 0, 1);
    const to = new Date();
    to.setTime(to.getTime() + 24 * 3600 * 1000);

    return API
      .get('analatics', '/analytics', {
        queryStringParameters: {
          from: from.toISOString().substring(0, 10),
          to: to.toISOString().substring(0, 10)
        }
      })
      .then(data => {
        if (data.success) {
          this.analytics = data.analysis;
        }
      })
      .catch(console.error);
  }
}
```

А.4 Програмний код компоненту створення та редагування вендингу

```
import { Component, ElementRef, OnDestroy, ViewChild, ViewEncapsulation } from '@angular/core';
import { FormArray, FormBuilder, FormControl, FormGroup } from '@angular/forms';
import { Location } from '@angular/common';
import { DataSource } from '@angular/cdk/collections';
import { MatSnackBar } from '@angular/material/snack-bar';
import { MatPaginator } from '@angular/material/paginator';
import { MatSort } from '@angular/material/sort';

import { BehaviorSubject, merge, Observable, Subject } from 'rxjs';
import { map, startWith, takeUntil } from 'rxjs/operators';

import { fuseAnimations } from '@fuse/animations';
import { FuseUtils } from '@fuse/utils';

import { VendingMachineService } from './machine.service';
import { Machine } from './machine.model';

@Component({
  selector: 'vending-machine',
  templateUrl: './machine.component.html',
  styleUrls: ['./machine.component.scss'],
  animations: fuseAnimations,
  encapsulation: ViewEncapsulation.None
})
export class VendingMachineComponent implements OnDestroy {
  productsDataSource: FilesDataSource | null;
  productsDisplayedColumns = ['id', 'name', 'price', 'local_quantity', 'active', 'remove'];

  @ViewChild(MatPaginator, {static: true})
  paginator: MatPaginator;

  @ViewChild(MatSort, {static: true})
  sort: MatSort;

  @ViewChild('filter', {static: true})
  filter: ElementRef;

  machine: Machine;
  products: any[];
  pageType: string;
  machineForm: FormGroup;
  productsForm: FormArray;
  filteredStores: Observable<any[]>;
  filteredProduct: Observable<any[]>;
  productControl: FormControl;

  // Private
```

```
  private _unsubscribeAll: Subject<any>;

  /**
   * Constructor
   *
   * @param {VendingMachineService} _machineService
   * @param {FormBuilder} _formBuilder
   * @param {Location} _location
   * @param {MatSnackBar} _matSnackBar
   */
  constructor(
    private _machineService: VendingMachineService,
    private _formBuilder: FormBuilder,
    private _location: Location,
    private _matSnackBar: MatSnackBar
  ) {
    // Set the default
    this.machine = new Machine();
    this.productsForm = new FormArray([]);
    this.productControl = new FormControl('');

    // Set the private defaults
    this._unsubscribeAll = new Subject();
  }

  // -----------------------------------------------------------------------------------------------------
  // @ Lifecycle hooks
  // -----------------------------------------------------------------------------------------------------

  /**
   * On init
   */
  ngOnInit(): void {
    // Subscribe to update product on changes
    this._machineService.onMachineChanged
      .pipe(takeUntil(this._unsubscribeAll))
      .subscribe(machine => {

        if (machine) {
          this.machine = new Machine(machine);
          this.pageType = 'edit';
        } else {
          this.pageType = 'new';
          this.machine = new Machine();
        }

        this.machineForm = this.createMachineForm();
        this.filteredStores = this.machineForm.get('store').valueChanges
          .pipe(
            startWith(''),
            map(store => store ? this._filterStore(store) : this._machineService.stores.slice())
          );
      });

    this.productsDataSource = new FilesDataSource(this.productsForm, this.paginator, this.sort);
    this._machineService.onMachineProductsChanged
      .pipe(takeUntil(this._unsubscribeAll))
      .subscribe(products => {
        setTimeout(() => {
          products.forEach((product, index) => {
            this.productsForm.insert(index, new FormControl(product));
          });
        }, 0);
      });

    this.filteredProduct = this.productControl.valueChanges
      .pipe(
        startWith(''),
        map(product => product ? this._filterProduct(product) : this._machineService.products.slice())
      );
  }
```

```
/**
 * On destroy
 */
ngOnDestroy(): void {
  // Unsubscribe from all subscriptions
  this._unsubscribeAll.next();
  this._unsubscribeAll.complete();
}

// -----------------------------------------------------------------------------------------------------
// @ Public methods
// -----------------------------------------------------------------------------------------------------

/**
 * Create machine form
 *
 * @returns {FormGroup}
 */
createMachineForm(): FormGroup {
  return this._formBuilder.group({
    id: [this.machine.id],
    name: [this.machine.name],
    description: [this.machine.description],
    location: [this.machine.location],
    enabled: [this.machine.enabled],
    store: [this.machine.store]
  });
}

storeDisplayWith(value: any): string {
  if (value) {
    if (typeof value === 'string') {
      return value;
    }
    if (typeof value === 'object') {
      return value.name;
    }
  }
  return '';
}

addProduct(): void {
  const product = this.productControl.value;
  if (product) {
    const index = this.productsForm.controls.findIndex(x => x.value.id === product.id);
    if (index === -1) {
      this._machineService.addMachineProduct(this.machine, product)
        .then(data => {
          this.productsForm.push(new FormControl(data));
        })
        .catch(console.error);
    }
  }
}

productQuantityChange(event: any, productMachine: any): void {
  const index = this.productsForm.controls.findIndex(x => x.value.id === productMachine.id);
  if (index > -1) {
    let quantity = 0;
    try {
      quantity = parseInt(event.target.value);
    } catch (e) {

    }
    if (quantity <= productMachine.product.quantity) {
      productMachine.quantity = quantity;
    } else {
      productMachine.quantity = productMachine.product.quantity;
    }
```

```
    this._machineService.saveMachineProduct(this.machine, productMachine)
      .then((data) => {
        this.productsForm.at(index).setValue(data);
      })
      .catch(console.error);
  }
}

productRemove(event: any, productMachine: any): void {
  this._machineService.removeMachineProduct(productMachine)
    .then(() => {
      const index = this.productsForm.controls.findIndex(x => x.value.id === productMachine.id);
      if (index > -1) {
        this.productsForm.removeAt(index);
      }
    })
    .catch(console.error);
}

/**
 * Save machine
 */
saveMachine(): void {
  const data = this.machineForm.getRawValue();
  data._version = this.machine._version;
  data.updatedAt = new Date().toISOString();
  data.machineStoreId = data.store && data.store.id || null;
  delete data.store;

  this._machineService.saveMachine(data)
    .then(machine => {

      // Trigger the subscription with new data
      this._machineService.onMachineChanged.next(machine);

      // Show the success message
      this._matSnackBar.open('Machine saved', 'OK', {
        verticalPosition: 'top',
        duration: 2000
      });
    });
}

/**
 * Add machine
 */
addMachine(): void {
  const data = this.machineForm.getRawValue();
  data._version = this.machine._version;
  data.createdAt = data.updatedAt = data.updatedAt = new Date().toISOString();
  data.machineStoreId = data.store && data.store.id || null;
  delete data.store;

  this._machineService.addMachine(data)
    .then(() => {

      // Trigger the subscription with new data
      this._machineService.onMachineChanged.next(data);

      // Show the success message
      this._matSnackBar.open('Machine added', 'OK', {
        verticalPosition: 'top',
        duration: 2000
      });

      // Change the location with new one
      this._location.go('apps/vending/machines/' + this.machine.id);
    });
}

private _filterStore(value: string): any[] {
```

```
    if (value && typeof value === 'string') {
      const filterValue = value.toLowerCase();

      return this._machineService.stores.filter(store => {
        return store.name.toLowerCase().indexOf(filterValue) === 0;
      });
    }
    return [value];
  }

  private _filterProduct(value: string): any[] {
    if (value && typeof value === 'string') {
      const filterValue = value.toLowerCase();

      return this._machineService.products.filter(store => {
        return store.name.toLowerCase().includes(filterValue) ||
          store.id.toLowerCase().indexOf(filterValue) === 0;
      });
    }
    return [value];
  }
}

class FilesDataSource extends DataSource<any> {
  private _filterChange = new BehaviorSubject('');
  private _filteredDataChange = new BehaviorSubject('');

  /**
   * Constructor
   *
   * @param {FormArray} _productsForm
   * @param {MatPaginator} _matPaginator
   * @param {MatSort} _matSort
   */
  constructor(
    private _productsForm: FormArray,
    private _matPaginator: MatPaginator,
    private _matSort: MatSort
  ) {
    super();

    this.filteredData = this._productsForm.value;
  }

  /**
   * Connect function called by the table to retrieve one stream containing the data to render.
   *
   * @returns {Observable<any[]>}
   */
  connect(): Observable<any[]> {
    const displayDataChanges = [
      this._productsForm.valueChanges,
      this._matPaginator.page,
      this._filterChange,
      this._matSort.sortChange
    ];

    return merge(...displayDataChanges)
      .pipe(
        map(() => {
            let data = this._productsForm.value.slice();

            data = this.filterData(data);

            this.filteredData = [...data];

            data = this.sortData(data);

            // Grab the page's slice of data.
            const startIndex = this._matPaginator.pageIndex * this._matPaginator.pageSize;
            return data.splice(startIndex, this._matPaginator.pageSize);
```

```
        }
      ));
  }

  // -----------------------------------------------------------------------------------------------------
  // @ Accessors
  // -----------------------------------------------------------------------------------------------------

  // Filtered data
  get filteredData(): any {
    return this._filteredDataChange.value;
  }

  set filteredData(value: any) {
    this._filteredDataChange.next(value);
  }

  // Filter
  get filter(): string {
    return this._filterChange.value;
  }

  set filter(filter: string) {
    this._filterChange.next(filter);
  }

  // -----------------------------------------------------------------------------------------------------
  // @ Public methods
  // -----------------------------------------------------------------------------------------------------

  /**
   * Filter data
   *
   * @param data
   * @returns {any}
   */
  filterData(data): any {
    if (!this.filter) {
      return data;
    }
    return FuseUtils.filterArrayByString(data, this.filter);
  }

  /**
   * Sort data
   *
   * @param data
   * @returns {any[]}
   */
  sortData(data): any[] {
    if (!this._matSort.active || this._matSort.direction === '') {
      return data;
    }

    return data.sort((a, b) => {
      let propertyA: number | string = '';
      let propertyB: number | string = '';

      switch (this._matSort.active) {
        case 'id':
          [propertyA, propertyB] = [a.product.id, b.product.id];
          break;
        case 'name':
          [propertyA, propertyB] = [a.product.name, b.product.name];
          break;
        case 'price':
          [propertyA, propertyB] = [a.product.cost, b.product.cost];
          break;
        case 'local_quantity':
          [propertyA, propertyB] = [a.quantity, b.quantity];
          break;
```

```
        case 'quantity':
          [propertyA, propertyB] = [a.product.quantity, b.product.quantity];
          break;
        case 'active':
          [propertyA, propertyB] = [a.product.enabled, b.product.enabled];
          break;
      }

      const valueA = isNaN(+propertyA) ? propertyA : +propertyA;
      const valueB = isNaN(+propertyB) ? propertyB : +propertyB;

      return (valueA < valueB ? -1 : 1) * (this._matSort.direction === 'asc' ? 1 : -1);
    });
  }

  /**
   * Disconnect
   */
  disconnect(): void {
  }
}
```

А.5 Програмний код провайдер сервіса даних для вендингу

```
import { Injectable } from '@angular/core';
import { ActivatedRouteSnapshot, RouterStateSnapshot } from '@angular/router';
import { BehaviorSubject, Observable } from 'rxjs';

import { APIService, CreateMachineProductsInput } from '../../../../API.service';
import { FuseUtils } from '../../../../../@fuse/utils';

@Injectable()
export class VendingMachineService {
  routeParams: any;
  machine: any;
  machineProducts: any[];
  stores: any[];
  products: any[];
  onMachineChanged: BehaviorSubject<any>;
  onMachineProductsChanged: BehaviorSubject<any[]>;
  onStoresChanged: BehaviorSubject<any[]>;
  onProductsChanged: BehaviorSubject<any[]>;

  /**
   * Constructor
   *
   * @param {APIService} _api
   */
  constructor(
    private _api: APIService
  ) {
    // Set the defaults
    this.onMachineChanged = new BehaviorSubject<any>({});
    this.onMachineProductsChanged = new BehaviorSubject<any[]>([]);
    this.onStoresChanged = new BehaviorSubject<any[]>([]);
    this.onProductsChanged = new BehaviorSubject<any[]>([]);
  }

  /**
   * Resolver
   *
   * @param {ActivatedRouteSnapshot} route
   * @param {RouterStateSnapshot} state
   * @returns {Observable<any> | Promise<any> | any}
   */
  resolve(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable<any> | Promise<any> | any {
    this.routeParams = route.params;

    return new Promise((resolve, reject) => {
```

```
    Promise.all([
      this.getMachine(),
      this.getMachineProducts(),
      this.getStores(),
      this.getProducts()
    ]).then(
      () => {
        resolve();
      },
      reject
    );
  });
}

/**
 * Get machine
 *
 * @returns {Promise<any>}
 */
getMachine(): Promise<any> {
  return new Promise((resolve, reject) => {
    if (this.routeParams.id === 'new') {
      this.onMachineChanged.next(false);
      resolve(false);
    } else {
      this._api.GetMachine(this.routeParams.id)
        .then(response => {
          this.machine = response;
          this.onMachineChanged.next(this.machine);
          resolve(response);
        })
        .catch(reject);
    }
  });
}

getMachineProducts(): Promise<any> {
  return this._api
    .ListMachineProductss({machineId: {eq: this.routeParams.id}}, 1000)
    .then(response => {
      this.machineProducts = response.items;
      this.onMachineProductsChanged.next(this.machineProducts);

      return response;
    });
}

getProducts(): Promise<any> {
  return this._api.ListProducts(null, 1000)
    .then(response => {
      this.products = response.items;
      this.onProductsChanged.next(this.products);

      return response;
    });
}

getStores(): Promise<any> {
  return this._api.ListStores(null, 1000)
    .then(response => {
      this.stores = response.items;
      this.onStoresChanged.next(this.stores);

      return response;
    });
}

/**
 * Save machine
 *
 * @param machine
```

```
   * @returns {Promise<any>}
   */
  saveMachine(machine): Promise<any> {
    return this._api.UpdateMachine(machine);
  }

  /**
   * Add machine
   *
   * @param machine
   * @returns {Promise<any>}
   */
  addMachine(machine): Promise<any> {
    return this._api.CreateMachine(machine);
  }

  addMachineProduct(machine, product): Promise<any> {
    const id = FuseUtils.generateGUID();
    const machineId = machine.id;
    const productId = product.id;
    const quantity = product.local_quantity;

    const data: CreateMachineProductsInput = {id, machineId, productId, quantity};
    return this._api.CreateMachineProducts(data);
  }

  saveMachineProduct(machine, productMachine): Promise<any> {
    const machineId = machine.id;
    const productId = productMachine.productId;
    const quantity: any = productMachine.quantity;

    return this._api.UpdateMachineProducts({
      id: productMachine.id,
      quantity,
      _version: productMachine._version
    }, {
      machineId: {eq: machineId},
      productId: {eq: productId}
    });
  }

  removeMachineProduct(productMachine): Promise<any> {
    return this._api.DeleteMachineProducts({
      id: productMachine.id,
      _version: productMachine._version
    });
  }
}
```

А.6 Програмний код компоненту списку вендигів

```
import { Component, ElementRef, OnDestroy, OnInit, ViewChild, ViewEncapsulation } from '@angular/core';
import { MatSnackBar } from '@angular/material/snack-bar';
import { MatPaginator } from '@angular/material/paginator';
import { MatSort } from '@angular/material/sort';
import { DataSource } from '@angular/cdk/collections';
import { BehaviorSubject, fromEvent, merge, Observable, Subject } from 'rxjs';
import { debounceTime, distinctUntilChanged, map, takeUntil } from 'rxjs/operators';

import { fuseAnimations } from '@fuse/animations';
import { FuseUtils } from '@fuse/utils';

import { VendingMachinesService } from './machines.service';

@Component({
  selector: 'vending-machines',
  templateUrl: './machines.component.html',
  styleUrls: ['./machines.component.scss'],
  animations: fuseAnimations,
  encapsulation: ViewEncapsulation.None
```

```
})
export class VendingMachinesComponent implements OnInit, OnDestroy {
  dataSource: FilesDataSource | null;
  displayedColumns = ['id', 'image', 'name', 'store', 'location', 'active', 'remove'];

  @ViewChild(MatPaginator, {static: true})
  paginator: MatPaginator;

  @ViewChild(MatSort, {static: true})
  sort: MatSort;

  @ViewChild('filter', {static: true})
  filter: ElementRef;

  // Private
  private _unsubscribeAll: Subject<any>;

  constructor(
    private _machinesService: VendingMachinesService,
    private _matSnackBar: MatSnackBar
  ) {
    // Set the private defaults
    this._unsubscribeAll = new Subject();
  }

  // -----------------------------------------------------------------------------------------------------
  // @ Lifecycle hooks
  // -----------------------------------------------------------------------------------------------------

  /**
   * On init
   */
  ngOnInit(): void {
    this.dataSource = new FilesDataSource(this._machinesService, this.paginator, this.sort);

    fromEvent(this.filter.nativeElement, 'keyup')
      .pipe(
        takeUntil(this._unsubscribeAll),
        debounceTime(150),
        distinctUntilChanged()
      )
      .subscribe(() => {
        if (!this.dataSource) {
          return;
        }

        this.dataSource.filter = this.filter.nativeElement.value;
      });
  }

  /**
   * On destroy
   */
  ngOnDestroy(): void {
    this._unsubscribeAll.next();
    this._unsubscribeAll.complete();
  }

  // -----------------------------------------------------------------------------------------------------
  // @ Public methods
  // -----------------------------------------------------------------------------------------------------

  /**
   * Remove Machine
   */
  onRemove(event: any, machine: any): void {
    event.stopPropagation();

    Promise.all([
      this._machinesService.removeMachine(machine),
      // this._machinesService.removeMachineProducts({id: machine.id})
```

```
    ])
      .then(() => {
        // Show the success message
        this._matSnackBar.open('Machine removed', 'OK', {
          verticalPosition: 'top',
          duration: 2000
        });

        this.dataSource.filter = '';
      })
      .catch(console.error);
  }
}

export class FilesDataSource extends DataSource<any> {
  private _filterChange = new BehaviorSubject('');
  private _filteredDataChange = new BehaviorSubject('');

  /**
   * Constructor
   *
   * @param {VendingMachinesService} _machinesService
   * @param {MatPaginator} _matPaginator
   * @param {MatSort} _matSort
   */
  constructor(
    private _machinesService: VendingMachinesService,
    private _matPaginator: MatPaginator,
    private _matSort: MatSort
  ) {
    super();

    this.filteredData = this._machinesService.machines;
  }

  /**
   * Connect function called by the table to retrieve one stream containing the data to render.
   *
   * @returns {Observable<any[]>}
   */
  connect(): Observable<any[]> {
    const displayDataChanges = [
      this._machinesService.onMachinesChanged,
      this._matPaginator.page,
      this._filterChange,
      this._matSort.sortChange
    ];

    return merge(...displayDataChanges)
      .pipe(
        map(() => {
            let data = this._machinesService.machines.slice();

            data = this.filterData(data);

            this.filteredData = [...data];

            data = this.sortData(data);

            // Grab the page's slice of data.
            const startIndex = this._matPaginator.pageIndex * this._matPaginator.pageSize;
            return data.splice(startIndex, this._matPaginator.pageSize);
          }
        ));
  }

  // -----------------------------------------------------------------------------------------------------
  // @ Accessors
  // -----------------------------------------------------------------------------------------------------

  // Filtered data
```

```
get filteredData(): any {
  return this._filteredDataChange.value;
}

set filteredData(value: any) {
  this._filteredDataChange.next(value);
}

// Filter
get filter(): string {
  return this._filterChange.value;
}

set filter(filter: string) {
  this._filterChange.next(filter);
}

// -----------------------------------------------------------------------------------------------------
// @ Public methods
// -----------------------------------------------------------------------------------------------------

/**
 * Filter data
 *
 * @param data
 * @returns {any}
 */
filterData(data): any {
  if (!this.filter) {
    return data;
  }
  return FuseUtils.filterArrayByString(data, this.filter);
}

/**
 * Sort data
 *
 * @param data
 * @returns {any[]}
 */
sortData(data): any[] {
  if (!this._matSort.active || this._matSort.direction === '') {
    return data;
  }

  return data.sort((a, b) => {
    let propertyA: number | string = '';
    let propertyB: number | string = '';

    switch (this._matSort.active) {
      case 'id':
        [propertyA, propertyB] = [a.id, b.id];
        break;
      case 'name':
        [propertyA, propertyB] = [a.name, b.name];
        break;
      case 'store':
        [propertyA, propertyB] = [a.store && a.store.name || null, b.store && b.store.name || null];
        break;
      case 'active':
        [propertyA, propertyB] = [a.enabled, b.enabled];
        break;
    }

    const valueA = isNaN(+propertyA) ? propertyA : +propertyA;
    const valueB = isNaN(+propertyB) ? propertyB : +propertyB;

    return (valueA < valueB ? -1 : 1) * (this._matSort.direction === 'asc' ? 1 : -1);
  });
}
```

```
  /**
   * Disconnect
   */
  disconnect(): void {
  }
}
```

А.7 Програмний код провайдер сервісу даних для компонента списку вендингів

```
import { Injectable } from '@angular/core';
import { ActivatedRouteSnapshot, Resolve, RouterStateSnapshot } from '@angular/router';
import { BehaviorSubject, Observable } from 'rxjs';

import { APIService } from '../../../../API.service';

@Injectable()
export class VendingMachinesService implements Resolve<any> {
  machines: any[];
  onMachinesChanged: BehaviorSubject<any>;

  /**
   * Constructor
   *
   * @param {APIService} _api
   */
  constructor(
    private _api: APIService
  ) {
    // Set the defaults
    this.onMachinesChanged = new BehaviorSubject({});
  }

  /**
   * Resolver
   *
   * @param {ActivatedRouteSnapshot} route
   * @param {RouterStateSnapshot} state
   * @returns {Observable<any> | Promise<any> | any}
   */
  resolve(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable<any> | Promise<any> | any {
    return new Promise((resolve, reject) => {

      Promise.all([
        this.getMachines()
      ]).then(
        () => {
          resolve();
        },
        reject
      );
    });
  }

  /**
   * Get machines
   *
   * @returns {Promise<any>}
   */
  getMachines(): Promise<any> {
    return new Promise(async (resolve, reject) => {
      try {
        const data = await this._api.ListMachines(undefined, 1000);

        this.machines = data.items;
        this.onMachinesChanged.next(this.machines);
        resolve(data.items);
      } catch (e) {
        reject(e);
```

```
      }
    });
  }

  /**
   * Remove machine
   *
   * @param machine
   */
  removeMachine(machine: any): Promise<any> {
    return this._api.DeleteMachine({
      id: machine.id,
      _version: machine._version
    })
      .then(res => {
        // Remove this machine from table
        const index = this.machines.findIndex(x => x.id === machine.id);
        if (index > -1) {
          this.machines.splice(index, 1);
        }
        return res;
      });
  }
}
```

А.8 Програмний код компоненту створення та редагування продукту

```
import { Component, OnDestroy, OnInit, ViewEncapsulation } from '@angular/core';
import { Router } from '@angular/router';
import { FormBuilder, FormControl, FormGroup } from '@angular/forms';
import { Location } from '@angular/common';
import { MatSnackBar } from '@angular/material/snack-bar';
import { Observable, Subject } from 'rxjs';
import { map, startWith, takeUntil } from 'rxjs/operators';

import { fuseAnimations } from '@fuse/animations';

import { Product } from 'app/main/apps/vending/product/product.model';
import { VendingProductService } from 'app/main/apps/vending/product/product.service';

@Component({
  selector: 'e-commerce-product',
  templateUrl: './product.component.html',
  styleUrls: ['./product.component.scss'],
  encapsulation: ViewEncapsulation.None,
  animations: fuseAnimations
})
export class VendingProductComponent implements OnInit, OnDestroy {
  product: Product;
  pageType: string;
  productForm: FormGroup;
  machineControl: FormControl;
  filteredMachines: Observable<any[]>;

  // Private
  private _unsubscribeAll: Subject<any>;

  /**
   * Constructor
   *
   * @param {VendingProductService} _productService
   * @param {FormBuilder} _formBuilder
   * @param {Location} _location
   * @param {Router} _router
   * @param {MatSnackBar} _matSnackBar
   */
  constructor(
    private _productService: VendingProductService,
    private _formBuilder: FormBuilder,
    private _location: Location,
    private _router: Router,
    private _matSnackBar: MatSnackBar
  ) {
    // Set the default
    this.product = new Product();
    this.machineControl = new FormControl('');

    // Set the private defaults
    this._unsubscribeAll = new Subject();
  }

  // -----------------------------------------------------------------------------------------------------
  // @ Lifecycle hooks
  // -----------------------------------------------------------------------------------------------------

  /**
   * On init
   */
  ngOnInit(): void {
    // Subscribe to update product on changes
    this._productService.onProductChanged
      .pipe(takeUntil(this._unsubscribeAll))
      .subscribe(product => {

        if (product) {
```

```
        this.product = new Product(product);
        this.pageType = 'edit';
      } else {
        this.pageType = 'new';
        this.product = new Product();
      }

      this.productForm = this.createProductForm();
    });

  this._productService.onMachinesChanged
    .pipe(takeUntil(this._unsubscribeAll))
    .subscribe(machines => {
      this.filteredMachines = this.machineControl.valueChanges
        .pipe(
          takeUntil(this._unsubscribeAll),
          startWith(''),
          map(product => product ?
            this._filterMachine(product) :
            this._productService.machines
              .filter(machine => this.product.machines
                .findIndex(x => x.machineId === machine.id) === -1)
              .slice()
          )
        );
    });
}

/**
 * On destroy
 */
ngOnDestroy(): void {
  // Unsubscribe from all subscriptions
  this._unsubscribeAll.next();
  this._unsubscribeAll.complete();
}

// -----------------------------------------------------------------------------------------------------
// @ Public methods
// -----------------------------------------------------------------------------------------------------

/**
 * Create product form
 *
 * @returns {FormGroup}
 */
createProductForm(): FormGroup {
  return this._formBuilder.group({
    id: [this.product.id],
    code: [this.product.code], // sku
    name: [this.product.name],
    description: [this.product.description],
    cost: [this.product.cost],
    enabled: [this.product.enabled],
    quantity: [this.product.quantity]
  });
}

/**
 * Save product
 */
saveProduct(): void {
  const data = this.productForm.getRawValue();
  data.updatedAt = new Date().toISOString();
  data._version = this.product._version;

  this._productService.saveProduct(data)
    .then((product) => {

      // Trigger the subscription with new data
      this._productService.onProductChanged.next(product);
```

```
      // Show the success message
      this._matSnackBar.open('Product saved', 'OK', {
        verticalPosition: 'top',
        duration: 2000
      });
    });
}

/**
 * Add product
 */
addProduct(): void {
  const data = this.productForm.getRawValue();
  data.createdAt = data.updatedAt = new Date().toISOString();

  this._productService.addProduct(data)
    .then(() => {

      // Trigger the subscription with new data
      this._productService.onProductChanged.next(data);

      // Show the success message
      this._matSnackBar.open('Product added', 'OK', {
        verticalPosition: 'top',
        duration: 2000
      });

      // Change the location with new one
      this._location.go('apps/vending/products/' + this.product.id);
    });
}

onOpenMachine(machineProduct: any): void {
  this._router.navigate(['apps', 'vending', 'machines', machineProduct.machineId]);
}

onAddMachine(): void {
  const machine = this.machineControl.value;
  if (machine) {
    this._productService.addMachineProduct(machine, {
      id: this.product.id,
      quantity: 0
    })
      .then(data => {
        // Clean filter
        this.machineControl.setValue('');

        // Add machine to table
        this.product.machines.push(data);

        // Show the success message
        this._matSnackBar.open('Machine added', 'OK', {
          verticalPosition: 'top',
          duration: 2000
        });
      })
      .catch(console.error);
  }
}

onMachineRemove(event: Event, machineProduct: any): void {
  event.stopPropagation();

  this._productService.removeMachineProduct(machineProduct)
    .then(machineProduct => {
      // Clean filter
      this.machineControl.setValue('');

      // Remove machine form table
      const index = this.product.machines.findIndex(x => x.id === machineProduct.id);
```

```
      if (index > -1) {
        this.product.machines.splice(index, 1);
      }

      // Show the success message
      this._matSnackBar.open('Machine removed', 'OK', {
        verticalPosition: 'top',
        duration: 2000
      });
    })
    .catch((e) => console.error(e));
}

onMachineProductUpdate(event: any, machineProduct: any, input: any): void {
  let quantity = 0;
  try {
    quantity = parseInt(event.target.value);
  } catch (e) {

  }

  const productQuantity = this.productForm.get('quantity').value || 0;
  if (quantity > productQuantity) {
    quantity = productQuantity;
  }

  this._productService
    .saveMachineProduct({id: machineProduct.machineId}, {
      id: machineProduct.id,
      productId: machineProduct.productId,
      quantity,
      _version: machineProduct._version
    })
    .then(data => {
      Object.assign(machineProduct, data);
      input.value = quantity.toString();
    })
    .catch((e) => console.error(e));
}

machineDisplayWith(value: any): string {
  if (value) {
    if (typeof value === 'string') {
      return value;
    }
    if (typeof value === 'object') {
      return value.name;
    }
  }
  return '';
}

private _filterMachine(value: string): any[] {
  if (value && typeof value === 'string') {
    const filterValue = value.toLowerCase();

    return this._productService.machines.filter(machine => {
      return this.product.machines.findIndex(x => x.machineId === machine.id) === -1 &&
        (machine.name.toLowerCase().indexOf(filterValue) === 0 ||
          machine.id.toLowerCase().indexOf(filterValue) === 0 ||
          machine.location.toLowerCase().indexOf(filterValue) === 0);
    });
  }
  return [value];
}
}
```

А.9 Програмний код провайдер сервісу даних для компоненту товарів

```
import { Injectable } from '@angular/core';
import { ActivatedRouteSnapshot, Resolve, RouterStateSnapshot } from '@angular/router';
import { BehaviorSubject, Observable } from 'rxjs';

import { APIService, CreateMachineProductsInput } from '../../../../API.service';
import { FuseUtils } from '../../../../../@fuse/utils';

@Injectable()
export class VendingProductService implements Resolve<any> {
  routeParams: any;
  product: any;
  machines: any[];
  onProductChanged: BehaviorSubject<any>;
  onMachinesChanged: BehaviorSubject<any[]>;

  /**
   * Constructor
   *
   * @param {APIService} _api
   */
  constructor(
    private _api: APIService
  ) {
    // Set the defaults
    this.onProductChanged = new BehaviorSubject({});
    this.onMachinesChanged = new BehaviorSubject<any[]>([]);
  }

  /**
   * Resolver
   *
   * @param {ActivatedRouteSnapshot} route
   * @param {RouterStateSnapshot} state
   * @returns {Observable<any> | Promise<any> | any}
   */
  resolve(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable<any> | Promise<any> | any {
    this.routeParams = route.params;

    return new Promise((resolve, reject) => {

      Promise.all([
        this.getProduct(),
        this.getMachines(),
      ]).then(
        () => {
          resolve();
        },
        reject
      );
    });
  }

  /**
   * Get product
   *
   * @returns {Promise<any>}
   */
  getProduct(): Promise<any> {
    return new Promise((resolve, reject) => {
      if (this.routeParams.id === 'new') {
        this.onProductChanged.next(false);
        resolve(false);
      } else {
        this._api.GetProduct(this.routeParams.id)
          .then(response => {
            const machines = response.machines && response.machines.items || [];
            const ids = Array.from(new Set(machines.map(x => x.machineId)));
            const promises: Promise<any>[] = ids.map(id => this._api.GetMachine(id).catch(e => console.error(e)));
            promises.unshift(Promise.resolve(response));

            return Promise.all(promises);
```

```
        })
        .then(data => {
          const [response, ...machines] = data;

          response.machines.items.forEach(item => {
            const index = machines.findIndex(x => x.id === item.machineId);
            if (index > -1) {
              item.machine = machines[index];
            }
          });

          return response;
        })
        .then(response => {
          this.product = response;
          this.onProductChanged.next(this.product);
          resolve(response);
        })
        .catch(reject);
    }
  });
}

/**
 * Get list of machines
 *
 * @returns {Promise<any>}
 */
getMachines(): Promise<any> {
  return this._api.ListMachines(null, 1000)
    .then(response => {
      this.machines = response.items;
      this.onMachinesChanged.next(this.machines);

      return response;
    });
}

/**
 * Save product
 *
 * @param product
 * @returns {Promise<any>}
 */
saveProduct(product): Promise<any> {
  return this._api.UpdateProduct(product);
}

/**
 * Add product
 *
 * @param product
 * @returns {Promise<any>}
 */
addProduct(product): Promise<any> {
  return this._api.CreateProduct(product);
}

addMachineProduct(machine, product): Promise<any> {
  const id = FuseUtils.generateGUID();
  const machineId = machine.id;
  const productId = product.id;
  const quantity = product.quantity;

  const data: CreateMachineProductsInput = {id, machineId, productId, quantity};
  return this._api.CreateMachineProducts(data);
}

saveMachineProduct(machine, productMachine): Promise<any> {
  const machineId = machine.id;
  const productId = productMachine.productId;
```

```
    const quantity: any = productMachine.quantity;

    return this._api.UpdateMachineProducts({
      id: productMachine.id,
      quantity,
      _version: productMachine._version
    }, {
      machineId: {eq: machineId},
      productId: {eq: productId}
    });
  }

  removeMachineProduct(productMachine): Promise<any> {
    return this._api.DeleteMachineProducts({
      id: productMachine.id,
      _version: productMachine._version
    });
  }
}
```

А.10 Програмний код компоненту списків продуктів

```
import { Component, ElementRef, OnDestroy, OnInit, ViewChild, ViewEncapsulation } from '@angular/core';
import { MatSnackBar } from '@angular/material/snack-bar';
import { MatPaginator } from '@angular/material/paginator';
import { MatSort } from '@angular/material/sort';
import { DataSource } from '@angular/cdk/collections';
import { BehaviorSubject, fromEvent, merge, Observable, Subject } from 'rxjs';
import { debounceTime, distinctUntilChanged, map, takeUntil } from 'rxjs/operators';

import { fuseAnimations } from '@fuse/animations';
import { FuseUtils } from '@fuse/utils';

import { VendingProductsService } from 'app/main/apps/vending/products/products.service';
import { Product } from '../product/product.model';

@Component({
  selector: 'vending-products',
  templateUrl: './products.component.html',
  styleUrls: ['./products.component.scss'],
  animations: fuseAnimations,
  encapsulation: ViewEncapsulation.None
})
export class VendingProductsComponent implements OnInit, OnDestroy {
  dataSource: FilesDataSource | null;
  displayedColumns = ['id', 'name', 'price', 'quantity', 'active', 'remove'];

  @ViewChild(MatPaginator, {static: true})
  paginator: MatPaginator;

  @ViewChild(MatSort, {static: true})
  sort: MatSort;

  @ViewChild('filter', {static: true})
  filter: ElementRef;

  // Private
  private _unsubscribeAll: Subject<any>;

  constructor(
    private _productsService: VendingProductsService,
    private _matSnackBar: MatSnackBar
  ) {
    // Set the private defaults
    this._unsubscribeAll = new Subject();
  }

  // -----------------------------------------------------------------------------------------------------
  // @ Lifecycle hooks
  // -----------------------------------------------------------------------------------------------------
```

```
  /**
   * On init
   */
  ngOnInit(): void {
    this.dataSource = new FilesDataSource(this._productsService, this.paginator, this.sort);

    fromEvent(this.filter.nativeElement, 'keyup')
      .pipe(
        takeUntil(this._unsubscribeAll),
        debounceTime(150),
        distinctUntilChanged()
      )
      .subscribe(() => {
        if (!this.dataSource) {
          return;
        }

        this.dataSource.filter = this.filter.nativeElement.value;
      });
  }

  /**
   * On destroy
   */
  ngOnDestroy(): void {
    this._unsubscribeAll.next();
    this._unsubscribeAll.complete();
  }

  // -----------------------------------------------------------------------------------------------------
  // @ Public methods
  // -----------------------------------------------------------------------------------------------------

  /**
   * Remove Product
   */
  onRemove(event: any, product: Product): void {
    event.stopPropagation();

    this._productsService.removeProduct({id: product.id, _version: product._version})
      .then(() => {
        // Show the success message
        this._matSnackBar.open('Product removed', 'OK', {
          verticalPosition: 'top',
          duration: 2000
        });

        this.dataSource.filter = '';
      })
      .catch(console.error);
  }
}

export class FilesDataSource extends DataSource<any> {
  private _filterChange = new BehaviorSubject('');
  private _filteredDataChange = new BehaviorSubject('');

  /**
   * Constructor
   *
   * @param {VendingProductsService} _productsService
   * @param {MatPaginator} _matPaginator
   * @param {MatSort} _matSort
   */
  constructor(
    private _productsService: VendingProductsService,
    private _matPaginator: MatPaginator,
    private _matSort: MatSort
  ) {
    super();
```

```
  this.filteredData = this._productsService.products;
}

/**
 * Connect function called by the table to retrieve one stream containing the data to render.
 *
 * @returns {Observable<any[]>}
 */
connect(): Observable<any[]> {
  const displayDataChanges = [
    this._productsService.onProductsChanged,
    this._matPaginator.page,
    this._filterChange,
    this._matSort.sortChange
  ];

  return merge(...displayDataChanges)
    .pipe(
      map(() => {
          let data = this._productsService.products.slice();

          data = this.filterData(data);

          this.filteredData = [...data];

          data = this.sortData(data);

          // Grab the page's slice of data.
          const startIndex = this._matPaginator.pageIndex * this._matPaginator.pageSize;
          return data.splice(startIndex, this._matPaginator.pageSize);
        }
      ));
}

// -----------------------------------------------------------------------------------------------------
// @ Accessors
// -----------------------------------------------------------------------------------------------------

// Filtered data
get filteredData(): any {
  return this._filteredDataChange.value;
}

set filteredData(value: any) {
  this._filteredDataChange.next(value);
}

// Filter
get filter(): string {
  return this._filterChange.value;
}

set filter(filter: string) {
  this._filterChange.next(filter);
}

// -----------------------------------------------------------------------------------------------------
// @ Public methods
// -----------------------------------------------------------------------------------------------------

/**
 * Filter data
 *
 * @param data
 * @returns {any}
 */
filterData(data): any {
  if (!this.filter) {
    return data;
  }
```

```
    return FuseUtils.filterArrayByString(data, this.filter);
  }

  /**
   * Sort data
   *
   * @param data
   * @returns {any[]}
   */
  sortData(data): any[] {
    if (!this._matSort.active || this._matSort.direction === '') {
      return data;
    }

    return data.sort((a, b) => {
      let propertyA: number | string = '';
      let propertyB: number | string = '';

      switch (this._matSort.active) {
        case 'id':
          [propertyA, propertyB] = [a.id, b.id];
          break;
        case 'name':
          [propertyA, propertyB] = [a.name, b.name];
          break;
        case 'price':
          [propertyA, propertyB] = [a.cost, b.cost];
          break;
        case 'quantity':
          [propertyA, propertyB] = [a.quantity, b.quantity];
          break;
        case 'active':
          [propertyA, propertyB] = [a.enabled, b.enabled];
          break;
      }

      const valueA = isNaN(+propertyA) ? propertyA : +propertyA;
      const valueB = isNaN(+propertyB) ? propertyB : +propertyB;

      return (valueA < valueB ? -1 : 1) * (this._matSort.direction === 'asc' ? 1 : -1);
    });
  }

  /**
   * Disconnect
   */
  disconnect(): void {
  }
}
```

А.11 Програмний код провайдер сервісу даних для компоненту списків продуктів

```
import { Injectable } from '@angular/core';
import { ActivatedRouteSnapshot, Resolve, RouterStateSnapshot } from '@angular/router';
import { BehaviorSubject, Observable } from 'rxjs';

import { APIService } from '../../../../API.service';

@Injectable()
export class VendingProductsService implements Resolve<any> {
  products: any[];
  onProductsChanged: BehaviorSubject<any>;

  /**
   * Constructor
   *
   * @param _api
   */
  constructor(
```

```
    private _api: APIService
  ) {
    // Set the defaults
    this.onProductsChanged = new BehaviorSubject({});
  }

  /**
   * Resolver
   *
   * @param {ActivatedRouteSnapshot} route
   * @param {RouterStateSnapshot} state
   * @returns {Observable<any> | Promise<any> | any}
   */
  resolve(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable<any> | Promise<any> | any {
    return new Promise((resolve, reject) => {

      Promise.all([
        this.getProducts()
      ]).then(
        () => {
          resolve();
        },
        reject
      );
    });
  }

  /**
   * Get products
   *
   * @returns {Promise<any>}
   */
  getProducts(): Promise<any> {
    return new Promise(async (resolve, reject) => {
      try {
        const data = await this._api.ListProducts(undefined, 100);

        this.products = data.items;
        this.onProductsChanged.next(this.products);
        resolve(data.items);
      } catch (e) {
        reject(e);
      }
    });
  }

  /**
   * Remove product
   *
   * @param product
   */
  removeProduct(product: any): Promise<any> {
    return this._api.DeleteProduct(product)
      .then(res => {
        // Remove this product from table
        const index = this.products.findIndex(x => x.id === product.id);
        if (index > -1) {
          this.products.splice(index, 1);
        }

        return res;
      });
  }
}
```

А.12 Програмний код компоненту створення та редагування магазинів або точок

```
import { Component, OnDestroy, OnInit, ViewEncapsulation } from '@angular/core';
import { Form, FormBuilder, FormControl, FormGroup } from '@angular/forms';
```

```
import { Location } from '@angular/common';
import { MatSnackBar } from '@angular/material/snack-bar';
import { Observable, Subject } from 'rxjs';
import { map, startWith, takeUntil } from 'rxjs/operators';

import { fuseAnimations } from '@fuse/animations';

import { VendingStoreService } from './store.service';
import { Store } from './store.model';
import { Router } from '@angular/router';
import { User } from '../../user-pool/user/user.model';

@Component({
  selector: 'vending-store',
  templateUrl: './store.component.html',
  styleUrls: ['./store.component.scss'],
  animations: fuseAnimations,
  encapsulation: ViewEncapsulation.None
})
export class VendingStoreComponent implements OnInit, OnDestroy {
  store: Store;
  machines: any[];
  employees: any[];
  users: User[];
  pageType: string;
  storeForm: FormGroup;
  machineControl: FormControl;
  employerControl: FormControl;
  filteredMachines: Observable<any[]>;
  filteredEmployees: Observable<User[]>;

  // Private
  private _unsubscribeAll: Subject<any>;

  /**
   * Constructor
   *
   * @param {VendingStoreService} _storeService
   * @param {FormBuilder} _formBuilder
   * @param {Location} _location
   * @param {Router} _router
   * @param {MatSnackBar} _matSnackBar
   */
  constructor(
    private _storeService: VendingStoreService,
    private _formBuilder: FormBuilder,
    private _location: Location,
    private _router: Router,
    private _matSnackBar: MatSnackBar
  ) {
    // Set the default
    this.store = new Store();
    this.machineControl = new FormControl('');
    this.employerControl = new FormControl('');

    // Set the private defaults
    this._unsubscribeAll = new Subject();
  }

  // -----------------------------------------------------------------------------------------------------
  // @ Lifecycle hooks
  // -----------------------------------------------------------------------------------------------------

  /**
   * On init
   */
  ngOnInit(): void {
    // Subscribe to update store on changes
    this._storeService.onStoreChanged
      .pipe(takeUntil(this._unsubscribeAll))
      .subscribe(store => {
```

```
      if (store) {
        this.store = new Store(store);
        this.pageType = 'edit';
      } else {
        this.pageType = 'new';
        this.store = new Store();
      }

      this.storeForm = this.createStoreForm();
    });

  // Subscribe to update machines on changes
  this._storeService.onMachinesChanged
    .pipe(takeUntil(this._unsubscribeAll))
    .subscribe(machines => {

      // Subscribe to updated machines for filter
      this.filteredMachines = this.machineControl.valueChanges
        .pipe(
          takeUntil(this._unsubscribeAll),
          startWith(''),
          map(product => product ? this._filterMachine(product) : this._storeService.machines.slice())
        );
    });

  this._storeService.onUsersChanged
    .pipe(takeUntil(this._unsubscribeAll))
    .subscribe(users => {
      this.users = users;

      // Subscribe to updated users for filter
      this.filteredEmployees = this.machineControl.valueChanges
        .pipe(
          takeUntil(this._unsubscribeAll),
          startWith(''),
          map(user => user ? this._filterUser(user) : this.users.slice())
        );
    });

  // Subscribe to update employees on changes
  this._storeService.onEmployeesChanged
    .pipe(takeUntil(this._unsubscribeAll))
    .subscribe(employees => {
      this.employees = employees;
    });
}

/**
 * On destroy
 */
ngOnDestroy(): void {
  // Unsubscribe from all subscriptions
  this._unsubscribeAll.next();
  this._unsubscribeAll.complete();
}

// -----------------------------------------------------------------------------------------------------
// @ Public methods
// -----------------------------------------------------------------------------------------------------

/**
 * Create store form
 *
 * @returns {FormGroup}
 */
createStoreForm(): FormGroup {
  return this._formBuilder.group({
    id: [this.store.id],
    name: [this.store.name],
    description: [this.store.description],
```

```
    location_type: [this.store.location_type],
    location: [this.store.location],
    enabled: [this.store.enabled],
  });
}

/**
 * Save store
 */
onSaveStore(): void {
  const data = this.storeForm.getRawValue();
  data.updatedAt = new Date().toISOString();
  data._version = this.store._version;

  this._storeService.saveStore(data)
    .then((product) => {
      // Trigger the subscription with new data
      this._storeService.onStoreChanged.next(product);

      // Show the success message
      this._matSnackBar.open('Store saved', 'OK', {
        verticalPosition: 'top',
        duration: 2000
      });
    });
}

/**
 * Add store
 */
onAddStore(): void {
  const data = this.storeForm.getRawValue();
  data.createdAt = data.updatedAt = new Date().toISOString();

  this._storeService.addStore(data)
    .then(() => {
      // Trigger the subscription with new data
      this._storeService.onStoreChanged.next(data);

      // Show the success message
      this._matSnackBar.open('Store added', 'OK', {
        verticalPosition: 'top',
        duration: 2000
      });

      // Change the location with new one
      this._location.go('apps/vending/stores/' + this.store.id);
    })
    .catch((e) => console.error(e));
}

onOpenMachine(machine: any): void {
  this._router.navigate(['apps', 'vending', 'machines', machine.id]);
}

onAddMachine(): void {
  const machine = this.machineControl.value;
  if (machine) {
    this._storeService.addMachine(this.store, machine)
      .then(data => {
        // Clean filter
        this.machineControl.setValue('');

        // Add machine to table
        this.store.machines.push(data);

        // Show the success message
        this._matSnackBar.open('Machine added', 'OK', {
          verticalPosition: 'top',
          duration: 2000
        });
```

```
      })
      .catch(console.error);
  }
}

onMachineRemove(event: Event, machine: any): void {
  event.stopPropagation();

  this._storeService.removeMachine(machine)
    .then(machine => {
      if (machine.store == null) {
        // Clean filter
        this.machineControl.setValue('');

        // Remove machine form table
        const index = this.store.machines.findIndex(x => x.id === machine.id);
        if (index > -1) {
          this.store.machines.splice(index, 1);
        }

        // Show the success message
        this._matSnackBar.open('Machine removed', 'OK', {
          verticalPosition: 'top',
          duration: 2000
        });
      }
    })
    .catch((e) => console.error(e));
}

onAddEmployer(): void {
  const value = this.employerControl.value;
  if (value) {

    const arr = [];
    if (value.name) {
      arr.push(value.name);
    }
    if (value.email) {
      arr.push(value.email);
    }
    const name = arr.join(', ');

    this._storeService.addEmployer({
      username: value.username,
      name: name,
      enabled: value.enabled
    })
      .then((data) => {
        // Clean filter
        this.employerControl.setValue('');

        this.employees.push(data);

        // Show the success message
        this._matSnackBar.open('Employer removed', 'OK', {
          verticalPosition: 'top',
          duration: 2000
        });
      })
      .catch((e) => console.error(e));
  }
}

onEmployerRemove(event: Event, employer: any): void {
  event.stopPropagation();

  this._storeService.removeEmployer(employer)
    .then(() => {
      // Clean filter
      this.employerControl.setValue('');
```

```
      // Remove user form table
      const index = this.employees.findIndex(x => x.id === employer.id);
      if (index > -1) {
        this.employees.splice(index, 1);
      }

      // Show the success message
      this._matSnackBar.open('Employer removed', 'OK', {
        verticalPosition: 'top',
        duration: 2000
      });
    })
    .catch((e) => console.error(e));
}

machineDisplayWith(value: any): string {
  if (value) {
    if (typeof value === 'string') {
      return value;
    }
    if (typeof value === 'object') {
      return value.name;
    }
  }
  return '';
}

employerDisplayWith(value: any): string {
  if (value) {
    if (typeof value === 'string') {
      return value;
    }
    if (typeof value === 'object') {
      const arr = [];
      if (value.name) {
        arr.push(value.name);
      }
      if (value.email) {
        arr.push(value.email);
      }
      return arr.join(', ');
    }
  }
  return '';
}

private _filterMachine(value: string): any[] {
  if (value && typeof value === 'string') {
    const filterValue = value.toLowerCase();

    return this._storeService.machines.filter(machine => {
      return machine.name.toLowerCase().indexOf(filterValue) === 0 ||
        machine.id.toLowerCase().indexOf(filterValue) === 0 ||
        machine.location.toLowerCase().indexOf(filterValue) === 0;
    });
  }
  return [value];
}

private _filterUser(value: string): any[] {
  if (value && typeof value === 'string') {
    const filterValue = value.toLowerCase();

    return this.users.filter(user => {
      return user.name.toLowerCase().indexOf(filterValue) === 0 ||
        user.username.toLowerCase().indexOf(filterValue) === 0 ||
        user.email.toLowerCase().indexOf(filterValue) === 0;
    });
  }
  return [value];
```

```
  }
}
```

А.13 Програмний код провайдер сервісу даних для компоненту магазинів або точок

```
import { Injectable } from '@angular/core';
import { ActivatedRouteSnapshot, RouterStateSnapshot } from '@angular/router';
import { BehaviorSubject, Observable } from 'rxjs';

import { APIService } from '../../../../API.service';
import { AdminQueries } from '../../admin.queries';
import { User } from '../../user-pool/user/user.model';
import { FuseUtils } from '../../../../../@fuse/utils';

@Injectable()
export class VendingStoreService {
  routeParams: any;

  store: any;
  onStoreChanged: BehaviorSubject<any>;
  machines: any[];
  onMachinesChanged: BehaviorSubject<any[]>;
  employees: any[];
  onEmployeesChanged: BehaviorSubject<any[]>;
  users: User[];
  onUsersChanged: BehaviorSubject<User[]>;

  private _nextToken;

  /**
   * Constructor
   *
   * @param {APIService} _api
   */
  constructor(
    private _api: APIService
  ) {
    // Set the defaults
    this.onStoreChanged = new BehaviorSubject({});
    this.onMachinesChanged = new BehaviorSubject<any[]>([]);
    this.onEmployeesChanged = new BehaviorSubject<any[]>([]);
    this.onUsersChanged = new BehaviorSubject<User[]>([]);
  }

  /**
   * Resolver
   *
   * @param {ActivatedRouteSnapshot} route
   * @param {RouterStateSnapshot} state
   * @returns {Observable<any> | Promise<any> | any}
   */
  resolve(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable<any> | Promise<any> | any {
    this.routeParams = route.params;

    return new Promise((resolve, reject) => {

      Promise.all([
        this.getStore(),
        this.getMachines(),
        this.getStoreEmployees(),
        this.listUsers(),
      ]).then(
        () => {
          resolve();
        },
        reject
      );
    });
  }
```

```
/**
 * Get store
 *
 * @returns {Promise<any>}
 */
getStore(): Promise<any> {
  return new Promise((resolve, reject) => {
    if (this.routeParams.id === 'new') {
      this.onStoreChanged.next(false);
      resolve(false);
    } else {
      this._api.GetStore(this.routeParams.id)
        .then(response => {
          this.store = response;
          this.onStoreChanged.next(this.store);
          resolve(response);
        })
        .catch(reject);
    }
  });
}

/**
 * Get list of machines
 *
 * @returns {Promise<any>}
 */
getMachines(): Promise<any> {
  return this._api.ListMachines(null, 1000)
    .then(response => {
      this.machines = response.items.filter(x => x.store == null);
      this.onMachinesChanged.next(this.machines);

      return response;
    });
}

/**
 * Get list Users
 *
 * @returns {Promise<any>}
 */
listUsers() {
  const query = {
    'limit': 60,
    'token': this._nextToken
  };

  return AdminQueries
    .request('/listUsers', 'get', query)
    .then(res => {
      const {NextToken, ...rest} = res;
      this._nextToken = NextToken;

      this.users = rest.Users.map(x => new User(x));
      this.onUsersChanged.next(this.users);

      return rest;
    });
}

getStoreEmployees(): Promise<any> {
  return this._api.ListStoreEmployees({
    storeId: {eq: this.routeParams.id}
  }, 1000)
    .then(response => {
      this.employees = response.items;
      this.onEmployeesChanged.next(this.employees);

      return response;
```

```
    });
}

/**
 * Save store
 *
 * @param store
 * @returns {Promise<any>}
 */
saveStore(store): Promise<any> {
  return this._api.UpdateStore(store);
}

/**
 * Add store
 *
 * @param store
 * @returns {Promise<any>}
 */
addStore(store): Promise<any> {
  return this._api.CreateStore(store);
}

/**
 * Add machine to store
 *
 * @param store
 * @param machine
 * @returns {Promise<any>}
 */
addMachine(store, machine): Promise<any> {
  return this._api
    .UpdateMachine({
      id: machine.id,
      _version: machine._version,
      machineStoreId: store.id
    })
    .then(machine => {
      const index = this.machines.findIndex(x => x.id === machine.id);
      if (index > -1) {
        this.machines.splice(index, 1);
        this.onMachinesChanged.next(this.machines);
      }

      return machine;
    });
}

/**
 * Remove machine from store
 *
 * @param machine
 * @returns {Promise<any>}
 */
removeMachine(machine): Promise<any> {
  return this._api
    .UpdateMachine({
      id: machine.id,
      _version: machine._version,
      machineStoreId: null
    })
    .then(machine => {
      const index = this.machines.findIndex(x => x.id === machine.id);
      if (index === -1) {
        this.machines.push(machine);
        this.onMachinesChanged.next(this.machines);
      }

      return machine;
    });
}
```

```
  addEmployer(employer): Promise<any> {
    const id = FuseUtils.generateGUID();
    return this._api.GetEmployee(employer.username)
      .then(data => {
        if (data == null) {
          return this._api.CreateEmployee({
            id: employer.username,
            name: employer.name,
            enabled: employer.enabled
          });
        }
      })
      .then(() => {
        return this._api.CreateStoreEmployee({
          id,
          storeId: this.store.id,
          employeeId: employer.username
        });
      });
  }

  removeEmployer(employer): Promise<any> {
    return this._api.DeleteStoreEmployee({id: employer.id, _version: employer._version}, {
      storeId: {eq: this.store.id}
    });
  }
}
```

А.14 Програмний код компоненту списків магазинів або точок

```
import { Component, ElementRef, OnDestroy, OnInit, ViewChild, ViewEncapsulation } from '@angular/core';
import { MatSnackBar } from '@angular/material/snack-bar';
import { MatPaginator } from '@angular/material/paginator';
import { MatSort } from '@angular/material/sort';
import { DataSource } from '@angular/cdk/collections';
import { BehaviorSubject, fromEvent, merge, Observable, Subject } from 'rxjs';
import { debounceTime, distinctUntilChanged, map, takeUntil } from 'rxjs/operators';

import { fuseAnimations } from '@fuse/animations';
import { FuseUtils } from '@fuse/utils';

import { VendingStoresService } from './stores.service';

@Component({
  selector: 'vending-stores',
  templateUrl: './stores.component.html',
  styleUrls: ['./stores.component.scss'],
  animations: fuseAnimations,
  encapsulation: ViewEncapsulation.None
})
export class VendingStoresComponent implements OnInit, OnDestroy {
  dataSource: FilesDataSource | null;
  displayedColumns = ['id', 'name', 'location', 'active', 'remove'];

  @ViewChild(MatPaginator, {static: true})
  paginator: MatPaginator;

  @ViewChild(MatSort, {static: true})
  sort: MatSort;

  @ViewChild('filter', {static: true})
  filter: ElementRef;

  // Private
  private _unsubscribeAll: Subject<any>;

  constructor(
    private _storesService: VendingStoresService,
    private _matSnackBar: MatSnackBar
  ) {
```

```
    // Set the private defaults
    this._unsubscribeAll = new Subject();
  }

  // -----------------------------------------------------------------------------------------------------
  // @ Lifecycle hooks
  // -----------------------------------------------------------------------------------------------------

  /**
   * On init
   */
  ngOnInit(): void {
    this.dataSource = new FilesDataSource(this._storesService, this.paginator, this.sort);

    fromEvent(this.filter.nativeElement, 'keyup')
      .pipe(
        takeUntil(this._unsubscribeAll),
        debounceTime(150),
        distinctUntilChanged()
      )
      .subscribe(() => {
        if (!this.dataSource) {
          return;
        }

        this.dataSource.filter = this.filter.nativeElement.value;
      });
  }

  /**
   * On destroy
   */
  ngOnDestroy(): void {
    this._unsubscribeAll.next();
    this._unsubscribeAll.complete();
  }

  // -----------------------------------------------------------------------------------------------------
  // @ Public methods
  // -----------------------------------------------------------------------------------------------------

  /**
   * Remove Machine
   */
  onRemove(event: any, machine: any): void {
    event.stopPropagation();

    this._storesService.removeStores({id: machine.id})
      .then(() => {
        // Show the success message
        this._matSnackBar.open('Machine removed', 'OK', {
          verticalPosition: 'top',
          duration: 2000
        });

        this.dataSource.filter = '';
      })
      .catch(console.error);
  }
}

export class FilesDataSource extends DataSource<any> {
  private _filterChange = new BehaviorSubject('');
  private _filteredDataChange = new BehaviorSubject('');

  /**
   * Constructor
   *
   * @param {VendingStoresService} _storesService
   * @param {MatPaginator} _matPaginator
   * @param {MatSort} _matSort
```

```
 */
constructor(
  private _storesService: VendingStoresService,
  private _matPaginator: MatPaginator,
  private _matSort: MatSort
) {
  super();

  this.filteredData = this._storesService.stores;
}

/**
 * Connect function called by the table to retrieve one stream containing the data to render.
 *
 * @returns {Observable<any[]>}
 */
connect(): Observable<any[]> {
  const displayDataChanges = [
    this._storesService.onStoresChanged,
    this._matPaginator.page,
    this._filterChange,
    this._matSort.sortChange
  ];

  return merge(...displayDataChanges)
    .pipe(
      map(() => {
          let data = this._storesService.stores.slice();

          data = this.filterData(data);

          this.filteredData = [...data];

          data = this.sortData(data);

          // Grab the page's slice of data.
          const startIndex = this._matPaginator.pageIndex * this._matPaginator.pageSize;
          return data.splice(startIndex, this._matPaginator.pageSize);
        }
      ));
}

// -----------------------------------------------------------------------------------------------------
// @ Accessors
// -----------------------------------------------------------------------------------------------------

// Filtered data
get filteredData(): any {
  return this._filteredDataChange.value;
}

set filteredData(value: any) {
  this._filteredDataChange.next(value);
}

// Filter
get filter(): string {
  return this._filterChange.value;
}

set filter(filter: string) {
  this._filterChange.next(filter);
}

// -----------------------------------------------------------------------------------------------------
// @ Public methods
// -----------------------------------------------------------------------------------------------------

/**
 * Filter data
 *
```

```
   * @param data
   * @returns {any}
   */
  filterData(data): any {
    if (!this.filter) {
      return data;
    }
    return FuseUtils.filterArrayByString(data, this.filter);
  }

  /**
   * Sort data
   *
   * @param data
   * @returns {any[]}
   */
  sortData(data): any[] {
    if (!this._matSort.active || this._matSort.direction === '') {
      return data;
    }

    return data.sort((a, b) => {
      let propertyA: number | string = '';
      let propertyB: number | string = '';

      switch (this._matSort.active) {
        case 'id':
          [propertyA, propertyB] = [a.id, b.id];
          break;
        case 'name':
          [propertyA, propertyB] = [a.name, b.name];
          break;
        case 'active':
          [propertyA, propertyB] = [a.enabled, b.enabled];
          break;
      }

      const valueA = isNaN(+propertyA) ? propertyA : +propertyA;
      const valueB = isNaN(+propertyB) ? propertyB : +propertyB;

      return (valueA < valueB ? -1 : 1) * (this._matSort.direction === 'asc' ? 1 : -1);
    });
  }

  /**
   * Disconnect
   */
  disconnect(): void {
  }
}
```

А.15 Програмний код провайдер сервісу даних для компоненту списків магазинів або точок

```
import { Injectable } from '@angular/core';
import { ActivatedRouteSnapshot, Resolve, RouterStateSnapshot } from '@angular/router';
import { BehaviorSubject, Observable } from 'rxjs';

import { APIService } from '../../../../API.service';

@Injectable()
export class VendingStoresService implements Resolve<any> {
  stores: any[];
  onStoresChanged: BehaviorSubject<any>;

  /**
   * Constructor
   *
   * @param {APIService} _api
   */
```

```
  constructor(
    private _api: APIService
  ) {
    // Set the defaults
    this.onStoresChanged = new BehaviorSubject({});
  }

  /**
   * Resolver
   *
   * @param {ActivatedRouteSnapshot} route
   * @param {RouterStateSnapshot} state
   * @returns {Observable<any> | Promise<any> | any}
   */
  resolve(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable<any> | Promise<any> | any {
    return new Promise((resolve, reject) => {

      Promise.all([
        this.getStores()
      ]).then(
        () => {
          resolve();
        },
        reject
      );
    });
  }

  /**
   * Get stores
   *
   * @returns {Promise<any>}
   */
  getStores(): Promise<any> {
    return new Promise(async (resolve, reject) => {
      try {
        const data = await this._api.ListStores(undefined, 100);

        this.stores = data.items;
        this.onStoresChanged.next(this.stores);
        resolve(data.items);
      } catch (e) {
        reject(e);
      }
    });
  }

  /**
   * Remove store
   *
   * @param store
   */
  removeStores(store: any): Promise<any> {
    store.enabled = false;
    return this._api.DeleteStore(store).then(res => {
      // Remove this store from table
      const index = this.stores.findIndex(x => x.id === store.id);
      if (index > -1) {
        this.stores.splice(index, 1);
      }
      return res;
    });
  }
}
```

А.16 Програмний код сервісу прийому платежів

```
const express = require('express');
const bodyParser = require('body-parser');
const awsServerlessExpressMiddleware = require('aws-serverless-express/middleware');
const https = require('https');
```

```
const AWS = require('aws-sdk');
const urlParse = require('url').URL;
const appsyncUrl = process.env.API_PRODUCTS_GRAPHQLAPIENDPOINTOUTPUT;
const region = process.env.REGION;
const endpoint = new urlParse(appsyncUrl).hostname.toString();
const {CreatePayment, GetProduct, UpdateMachineProducts} = require('./query.js');
const apiKey = process.env.API_KEY;
const {v4} = require('uuid');

// declare a new express app
const app = express();
app.use(bodyParser.json());
app.use(awsServerlessExpressMiddleware.eventContext());

// Enable CORS for all methods
app.use(function (req, res, next) {
  res.header('Access-Control-Allow-Origin', '*');
  res.header('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content-Type, Accept');
  next();
});

app.put('/payments', async function (req, res) {
  const machineId = req.body.machineId;
  const productId = req.body.productId;

  const productResult = await graphRequest({
    query: GetProduct,
    operationName: "GetProduct",
    variables: {
      id: productId
    }
  });

  console.log("GetProduct", JSON.stringify(productResult, null, 2));

  if (productResult && productResult.data && productResult.data.getProduct) {
    const product = productResult.data.getProduct;
    const link = product.machines.items.find(x => x.machineId === machineId);

    const resultUpdate = await graphRequest({
      query: UpdateMachineProducts,
      operationName: "UpdateMachineProducts",
      variables: {
        input: {
          id: link.id,
          quantity: link.quantity - 1,
          _version: link._version,
        }
      }
    });

    console.log("UpdateMachineProducts", JSON.stringify(resultUpdate, null, 2));

    if (resultUpdate && resultUpdate.data && resultUpdate.data.updateMachineProducts) {
      const createResult = await graphRequest({
        query: CreatePayment,
        operationName: "CreatePayment",
        variables: {
          input: {
            id: v4(),
            storeId: link.machine && link.machine.store && link.machine.store.id || null,
            machineId,
            productId,
            cost: product.cost || 0,
            createdAt: new Date().toISOString()
          }
        }
      });

      console.log("CreatePayment", JSON.stringify(createResult, null, 2));
```

```
    if (createResult && createResult.data && createResult.data.createPayment) {
      return res.json({success: true, body: createResult.data});
    }
   }
  }

  return res.json({success: false});
});

app.listen(3000, function () {
  console.log('App started');
  console.log(`ENV: ${JSON.stringify(process.env, null, 2)}`);
});

// Export the app object. When executing the application local this does nothing. However,
// to port it to AWS Lambda we will create a wrapper around that will load the app from
// this file
module.exports = app;

const graphRequest = (body) => {
  const awsReq = new AWS.HttpRequest(appsyncUrl, region);
  awsReq.method = "POST";
  awsReq.headers.host = endpoint;
  awsReq.headers["Content-Type"] = "application/json";
  awsReq.body = JSON.stringify(body);

  if (apiKey) {
    awsReq.headers["x-api-key"] = apiKey;
  } else {
    const signer = new AWS.Signers.V4(awsReq, "appsync", true);
    signer.addAuthorization(AWS.config.credentials, AWS.util.date.getDate());
  }

  console.log(`AWS REQ: `, JSON.stringify(awsReq, null, 2));

  return new Promise((resolve, reject) => {
    const httpRequest = https.request({ ...awsReq, host: endpoint }, (result) => {
      result.on('data', (data) => {
        resolve(JSON.parse(data.toString()));
      });
    });

    httpRequest.write(awsReq.body);
    httpRequest.end();
  });
};
```

А.17 Програмний код сервісу аналітики

```
const express = require('express');
const bodyParser = require('body-parser');
const awsServerlessExpressMiddleware = require('aws-serverless-express/middleware');
const axios = require('axios');
const gql = require('graphql-tag');
const graphql = require('graphql');
const {ListPayments, ListStores, ListProducts} = require('./query');
const {print} = graphql;
const API_END_POINT = process.env.API_URL || process.env.API_PRODUCTS_GRAPHQLAPIENDPOINTOUTPUT;

// declare a new express app
const app = express();
app.use(bodyParser.json());
app.use(awsServerlessExpressMiddleware.eventContext());

// Enable CORS for all methods
app.use(function (req, res, next) {
  res.header('Access-Control-Allow-Origin', '*')
  res.header('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content-Type, Accept')
  next()
});
```

```
/*************************
 * get analytics method *
 ************************/
app.get('/analytics', async (req, res) => {
  const from = req.query.from;
  const to = req.query.to;

  try {
    const payments = await getPayments(from, to, undefined, []);
    if (payments && Array.isArray(payments)) {

      const total_cost = payments.reduce((acc, current) => acc += current.cost, 0);
      const total_payments = payments.length;

      const payments_by_date = groups(payments, (payment) => payment.createdAt.split('T')[0]);
      const payments_by_month = groups(Object.keys(payments_by_date), (rawDate) => new Date(rawDate).getMonth());

      let payments_by_months = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
      Object
        .keys(payments_by_month)
        .forEach(key => payments_by_month[key]
          .forEach(date => payments_by_months[key] += payments_by_date[date].length)
        );
      payments_by_months = payments_by_months.map(x => x / 1000);

      const now = new Date();
      const today = now.toISOString().substring(0, 10);
      const weeks = getWeek(now);

      const payments_today = payments_by_date.hasOwnProperty(today) ? payments_by_date[today].length : 0;
      const payments_week = weeks.map(date => payments_by_date[date] && payments_by_date[date].length || 0);

      let sales_today = 0;
      if (payments_today > 0) {
        sales_today= payments_by_date[today].reduce((prevValue, currentValue) => {
          prevValue += currentValue.cost;
          return prevValue;
        }, 0);
      }

      // ------------
      const stores = await getStores();
      const products = await getProducts();

      const store_payments_week = {};
      weeks
        .filter(date => payments_by_date[date])
        .forEach(date => {
          const payments = payments_by_date[date];
          payments.forEach(payment => {
            if (!store_payments_week.hasOwnProperty(payment.storeId)) {
              store_payments_week[payment.storeId] = [];
            }
            store_payments_week[payment.storeId].push(payment);
          });
        });

      // -----
      const s_payments_week = [];
      Object
        .keys(store_payments_week)
        .forEach(storeId => {
          const store = stores.find(x => x.id === storeId);

          const obj = {
            name: store.name,
            series: []
          };

          const productGroup = groups(store_payments_week[storeId], (payment) => payment.productId);
          obj.series = Object
```

```
          .keys(productGroup)
          .map(productId => {
            const product = products.find(x => x.id === productId);
            return {
              name: product.name,
              value: productGroup[productId].length
            };
          });

        s_payments_week.push(obj);
      });
    // ------------
    const sales_week = [];
    const products_week = {};

    weeks
      .filter(date => payments_by_date[date])
      .forEach(date => {
        const payments = payments_by_date[date];

        // -----
        const sum = payments.reduce((acc, current) => acc += current.cost, 0);
        sales_week.push({
          name: date,
          value: sum
        });
        // -----

        const productGroup = groups(payments, (payment) => payment.productId);
        Object
          .keys(productGroup)
          .forEach(productId => {
            const length = productGroup[productId].length;
            if (products_week.hasOwnProperty(productId)) {
              products_week[productId].value += length;
            } else {
              const product = products.find(x => x.id === productId);
              products_week[productId] = {
                name: product.name,
                value: length
              };
            }
          });
      });
    // ---

    return res.json({
      success: true,
      analysis: {
        total: {
          cost: total_cost,
          payments: total_payments
        },
        today: {
          payments_today,
          sales_today
        },
        payments_by_months,
        payments_week,
        sales_week,
        products_week: Object.values(products_week),
        store_payments_week: s_payments_week
      }
    });
  }
  return res.json({success: false});
 } catch (e) {
  console.log('ERROR', e);
  return res.json({success: false, error: e});
 }
});
```

```
app.listen(3000, function () {
  console.log('App started')
});

// Export the app object. When executing the application local this does nothing. However,
// to port it to AWS Lambda we will create a wrapper around that will load the app from
// this file
module.exports = app;

const groups = (data, keyFn) => {
  return data.reduce((groups, payment) => {
    const key = keyFn(payment);
    if (!groups[key]) {
      groups[key] = [];
    }
    groups[key].push(payment);
    return groups;
  }, {});
};

const getPayments = async (from, to, nextToken, array) => {
  const query = gql(ListPayments);
  const limit = 5000;
  const request = await axios({
    url: API_END_POINT,
    method: 'post',
    headers: {
      'x-api-key': process.env.API_KEY
    },
    data: {
      query: print(query),
      variables: {
        filter: {
          createdAt: {
            between: [from, to]
          }
        },
        limit,
        nextToken
      }
    }
  });

  const {data} = request.data;
  if (data && data.listPayments && Array.isArray(data.listPayments.items) && data.listPayments.items.length > 0) {
    array.push(...data.listPayments.items);

    const nextToken = data.listPayments.nextToken;
    if (nextToken) {
      return getPayments(from, to, nextToken, array);
    }
  }

  return array;
};

const getStores = async () => {
  const query = gql(ListStores);
  const request = await axios({
    url: API_END_POINT,
    method: 'post',
    headers: {
      'x-api-key': process.env.API_KEY
    },
    data: {
      query: print(query),
      variables: {
        limit: 5000
      }
    }
```

```
  });

  const {data} = request.data;
  if (data && data.listStores && Array.isArray(data.listStores.items) && data.listStores.items.length > 0) {
    return data.listStores.items;
  }
  return [];
};

const getProducts = async () => {
  const query = gql(ListProducts);
  const request = await axios({
    url: API_END_POINT,
    method: 'post',
    headers: {
      'x-api-key': process.env.API_KEY
    },
    data: {
      query: print(query),
      variables: {
        limit: 5000
      }
    }
  });

  const {data} = request.data;
  if (data && data.listProducts && Array.isArray(data.listProducts.items) && data.listProducts.items.length > 0) {
    return data.listProducts.items;
  }
  return [];
};

function getWeek(d) {
  d = new Date(d);
  const day = d.getDay(),
    diff = d.getDate() - day + (day == 0 ? -6 : 1); // adjust when day is sunday
  const begin = new Date(d.setDate(diff));

  const hours = -(new Date().getTimezoneOffset() / 60);
  const minutes = -(new Date().getTimezoneOffset() % 60);
  begin.setHours(hours, minutes, 0, 0);

  const DAY = 1000 * 60 * 60 * 24;
  const res = [];

  for (let i = 0; i < 7; i++) {
    const date = new Date();
    date.setTime(begin.getTime() + i*DAY);

    res.push(date.toISOString().substring(0, 10));
  }

  return res;
}
```

ДОДАТОК Б

Опис моделі GraphQL

```
type Store @model
  @auth(rules: [
    {allow: groups, provider: userPools, groups: ["Managers"], operations: [read, create, update, delete]},
    {allow: groups, provider: userPools, groups: ["Employees"], operations: [read], mutations: null},
    { allow: public, provider: apiKey, operations: [create, read] },
    { allow: public, provider: iam}
  ]) {
  id: ID!
  name: String!
  description: String
  location_type: String!
  location: String!
  createdAt: AWSDateTime!
  updatedAt: AWSDateTime!
  enabled: Boolean!
  machines: [Machine] @connection(name: "StoreMachines")
  employees: [StoreEmployee] @connection(keyName: "byStore", fields: ["id"])
}

type Machine @model
  @auth(rules: [
    {allow: groups, provider: userPools, groups: ["Managers"], operations: [read, create, update, delete]},
    {allow: groups, provider: userPools, groups: ["Employees"], operations: [read], mutations: null},
    { allow: public, provider: apiKey, operations: [create, read] },
    { allow: public, provider: iam}
  ]) {
  id: ID!
  name: String!
  description: String
  image: String
  location: String!
  createdAt: AWSDateTime!
  updatedAt: AWSDateTime!
  enabled: Boolean!
  store: Store @connection(name: "StoreMachines")
  products: [MachineProducts] @connection(keyName: "byMachine", fields: ["id"])
}

type MachineProducts @model
  @key(name: "byMachine", fields: ["machineId", "productId"])
  @key(name: "byProduct", fields: ["productId", "machineId"])
  @auth(rules: [
    {allow: groups, provider: userPools, groups: ["Managers"], operations: [read, create, update, delete]},
    {allow: groups, provider: userPools, groups: ["Employees"], operations: [read], mutations: null},
    { allow: public, provider: apiKey, operations: [create, read, update] },
    { allow: public, provider: iam}
  ]) {
  id: ID
  machineId: ID!
  productId: ID!
  machine: Machine @connection(fields: ["machineId"])
  product: Product @connection(fields: ["productId"])
  quantity: Int
}

type Product @model
  @auth(rules: [
```

```
    {allow: groups, provider: userPools, groups: ["Managers"], operations: [read, create, update, delete]},
    {allow: groups, provider: userPools, groups: ["Employees"], operations: [read], mutations: null},
    { allow: public, provider: apiKey, operations: [create, read] },
    { allow: public, provider: iam}
  ]) {
  id: ID!
  code: String!
  name: String!
  description: String
  image: String
  cost: Float
  createdAt: AWSDateTime!
  updatedAt: AWSDateTime!
  enabled: Boolean!
  quantity: Int
  machines: [MachineProducts] @connection(keyName: "byProduct", fields: ["id"])
}

type StoreEmployee @model
@key(name: "byStore", fields: ["storeId", "employeeId"])
@key(name: "byEmployee", fields: ["employeeId", "storeId"])
@auth(rules: [
  {allow: groups, provider: userPools, groups: ["Managers"], operations: [read, create, update, delete]},
  {allow: groups, provider: userPools, groups: ["Employees"], operations: [read], mutations: null},
  { allow: public, provider: apiKey, operations: [create, read] },
  { allow: public, provider: iam}
]) {
  id: ID!
  storeId: ID!
  employeeId: ID!
  store: Store @connection(fields: ["storeId"])
  employee: Employee @connection(fields: ["employeeId"])
}

type Employee @model
  @auth(rules: [
    {allow: groups, provider: userPools, groups: ["Managers"], operations: [read, create, update, delete]},
    {allow: groups, provider: userPools, groups: ["Employees"], operations: [read], mutations: null},
    { allow: public, provider: iam}
  ]) {
  id: ID!
  name: String!
  enabled: Boolean
  stores: [StoreEmployee] @connection(keyName: "byEmployee", fields: ["id"])
}

type Payment @model
@key(name: "byCreatedAt", fields: ["createdAt"])
  @auth(rules: [
    {allow: groups, provider: userPools, groups: ["Managers"], operations: [read, create], mutations: null},
    {allow: groups, provider: userPools, groups: ["Employees"], operations: [read], mutations: null},
    { allow: public, provider: apiKey, operations: [create, read] },
    { allow: public, provider: iam}
  ]) {
  id: ID!
  storeId: ID!
  machineId: ID!
  productId: ID!
  store: Store @connection(fields: ["storeId"])
  machine: Machine @connection(fields: ["machineId"])
  product: Product @connection(fields: ["productId"])
  cost: Float!
  createdAt: AWSDateTime!
}
```

ДОДАТОК В

Опис основних методів та класів для взаємодії бази даних та інтерфейсу користувача

Таблиця В.1 – Опис основних методів класу взаємодії з базою даних

Класс	Метод Параметри виклику Тип значення, що повертається	Опис дій
APIService	CreateStore CreateStoreInput input, ModelStoreConditionInput? condition Promise<CreateStore Mutation>	Метод створення нових магазинів або точок за вхідним параметром input та condition для умов запиту
APIService	UpdateStore UpdateStoreInput input, ModelStoreConditionInput? condition Promise<UpdateStoreMutation>	Метод оновлення даних про магазин за допомогою id магазина на версії документу
APIService	DeleteStore DeleteStoreInput input, ModelStoreConditionInput? condition Promise<DeleteStoreMutation>	Метод видалення даних про магазин за допомогою id магазина та версією документа

Продовження таблиці В.1

APIService	CreateMachine CreateMachineInput input, ModelMachineCondition condition Promise<CreateMachineMutation>	Метод створює новий вендинговий автомат за переданими даними до моделі
APIService	UpdateMachine UpdateMachineInput input, ModelMachineConditionInput condition Promise<UpdateMachineMutation>	Метод оновлення даних про вендинг на основі переданих даних до моделі
APIService	DeleteMachine DeleteMachineInput input, ModelMachineConditionInput condition Promise<DeleteMachineMutation>	Метод видалення вендингу по ід на версії
APIService	CreateMachineProducts CreateMachineProductsInput input, ModelMachineProductsConditionInput condition Promise<CreateMachineProductsMutation>	Метод створення зв’язку між машиною та продуктами за вхідними параметрами

Продовження таблиці В.1

APIService	UpdateMachineProducts UpdateMachineProductsInput input, ModelMachineProductsConditionInput condition Promise<UpdateMachineProductsMutation>	Метод оновлення даних в зв’язку товар – машина за вхідними параметрами
APIService	DeleteMachineProducts DeleteMachineProductsInput input, ModelMachineProductsConditionInput condition Promise<DeleteMachineProductsMutation>	Метод видалення зв’язку товар – машина за допомогою Id та версії документу
APIService	CreateProduct CreateProductInput input, ModelProductConditionInput condition Promise<CreateProductMutation>	Метод створення продукту за вхідними параметрами
APIService	UpdateProduct UpdateProductInput input, ModelProductConditionInput condition Promise<UpdateProductMutation>	Метод оновлення продукту за вхідними параметрами
APIService	DeleteProduct DeleteProductInput input, ModelProductConditionInput condition Promise<DeleteProductMutation>	Метод видалення продукта за вхідними параметрами

Продовження таблиці В.1

APIService	CreateStoreEmployee CreateStoreEmployeeInput input, ModelStoreEmployeeConditionInput condition Promise<CreateStoreEmployeeMutation>	Метод створення зв’язку між працівником та магазином або точкою за вхідними параметрами
APIService	DeleteStoreEmployee DeleteStoreEmployeeInput input, ModelStoreEmployeeConditionInput condition Promise<DeleteStoreEmployeeMutation>	Метод видалення зв’язку між працівником та магазином або точною за вхідними параметрами
APIService	CreateEmployee CreateEmployeeInput input, ModelEmployeeConditionInput condition Promise<CreateEmployeeMutation>	Метод створення працівника за вхідними параметрами
APIService	UpdateEmployee UpdateEmployeeInput input, ModelEmployeeConditionInput condition Promise<UpdateEmployeeMutation>	Метод оновлення даних працівника за вхідними параметрами

Продовження таблиці В.1

APIService	CreatePayment CreatePaymentInput input, ModelPaymentConditionInput condition Promise<CreatePaymentMutation>	Метод створення операції за вхідними параметрами
APIService	ListStores ModelStoreFilterInput filter, Number? limit, String? nextToken Promise<ListStoresQuery>	Метод отримання списку магазинів або точок за вхідними параметрами
APIService	GetStore String id Promise<GetStoreQuery>	Метод отримання магазину або точки за id магазину
APIService	ListMachines ModelMachineFilterInput filter, Number? Limit, String? nextToken Promise<ListMachinesQuery>	Метод отримання списку вендингів за вхідними параметрами
APIService	GetMachine String id Promise<GetMachineQuery>	Метод отримання вендингу за Id
APIService	ListProducts ModelProductFilterInput filter, Number? limit, String? nextToken Promise<ListProductsQuery>	Метод отримання списку продуктів за вхідними параметрами

Продовження таблиці В.1

APIService	GetProduct String id Promise<GetProductQuery>	Метод отримання певного товару за id
APIService	ListEmployees ModelEmployeeFilterInput filter, Number? limit, String? nextToken Promise<ListEmployeesQuery>	Метод отримання списку працівників за вхідними параметрами
APIService	GetEmployee String id Promise<GetEmployee Query>	Метод отримання працівника за допомого id

Таблиця В.2 – Опис основних методів класів модулю UI

Класс	Метод Параметри виклику Тип значення, що повертається	Опис дій
LoginComponent	ngOnInit - void	Метод ініціалізації компонента
LoginComponent	onSubmit - void	Метод валідації та відправки даних на авторизацію
RegisterComponent	ngOnInit - void	Метод ініціалізації компонента
RegisterComponent	ngOnDestroy - void	Метод деструкції компонента
RegisterComponent	onSubmit - void	Метод валідації та відправки даних на реєстрацію

Продовження таблиці В.2

MailConfirmComponent	ngOnInit - void	Метод ініціалізації компонента
MailConfirmComponent	onSubmit - void	Метод валідації та відправки даних на підтвердження пошти
ForgotPasswordComponent	ngOnInit - void	Метод ініціалізації компонента

Продовження таблиці В.2

ForgotPasswordComponent	onSubmit - void	Метод валідації та відправки даних по відновлення пароля
AnalyticsDashboardComponent	ngOnInit - void	Метод ініціалізації компонента
AnalyticsDashboardService	resolve - Promise<void>	Метод попередньої ініціалізації даних для компонента
AnalyticsDashboardService	getActiveProducts - Promise<void>	Метод запиту та обробки списку активних продуктів
AnalyticsDashboardService	getAnalytics - Promise<void>	Метод запиту та обробки аналітичних наборів
ProjectDashboardComponent	ngOnInit - void	Метод ініціалізації компонента
ProjectDashboardService	resolve - Promise<void>	Метод попередньої ініціалізації даних для компонента

Продовження таблиці В.2

ProjectDashboardService	getProjectData - Promise<void>	Метод запиту та обротки списку, вендингу та товарів
VendingMachineComponent	ngOnInit - void	Метод ініціалізації компонента
VendingMachineComponent	ngOnDestroy - void	Метод деструкції компонента
VendingMachineComponent	createMachineForm - FormGroup	Метод створення реактивної форми
VendingMachineComponent	storeDisplayWith value: any string	Метод для відображення значення у виборі магазина
VendingMachineComponent	addProduct - void	Метод створює зв’язок між вендингом та продуктом

Продовження таблиці В.2

VendingMachineComponent	productRemove event: any, productMachine: any void	Метод видаляє зв’язок між вендингом та продуктом
VendingMachineComponent	productQuantityChange event: any, productMachine: any void	Метод оновлює інформацію кількості продуктів у вендингу
VendingMachineComponent	saveMachine - void	Метод оновлення даних вендингу
VendingMachineComponent	addMachine - void	Метод створення вендингу
VendingMachineService	resolve route: ActivatedRouteSnapshot, state: RouterStateSnapshot Promise<any>	Метод попередньої ініціалізації даних для компонента
VendingMachineService	getMachine - Promise<any>	Метод отримання інформації про вендинг

Продовження таблиці В.2

VendingMachineService	getMachineProducts - Promise<any>	Метод отримання списку продуктів у вендинзі
VendingMachineService	getProducts - Promise<any>	Метод отримує список активних продуктів для подальшого додавання
VendingMachineService	getStores - Promise<any>	Метод отримує список активних магазинів для подальшого зв’язку вендинга з магазином
VendingMachineService	saveMachine machine: any Promise<any>	Метод створює та повертає ієрархічне представлення моделі машини
VendingMachineService	addMachine machine: any Promise<any>	Метод створює новий екземпляр моделі та повертає результат створення

Продовження таблиці В.2

VendingMachineService	addMachineProduct machine, product Promise<any>	Метод створює новий екземпляр моделі зв’язку вендингу з продукту
VendingMachineService	saveMachineProduct machine, productMachine Promise<any>	Метод оновлює кількість продуктів у зв’язку
VendingMachineService	removeMachineProduct productMachine Promise<any>	Метод видаляє зв’язок вендингу з продуктом
VendingMachinesComponent	ngOnInit - void	Метод ініціалізації компонента
VendingMachinesComponent	ngOnDestroy - void	Метод деструкції компонента
VendingMachinesComponent	onRemove event: any, machine: any void	Метод видалення вендингу

Продовження таблиці В.2

VendingMachinesService	resolve route: ActivatedRouteSnapshot, state: RouterStateSnapshot Promise<any>	Метод попередньої ініціалізації даних необхідних для компонента
VendingMachinesService	getMachines - Promise<any>	Метод отримання списку активних вендингів з бази даних
VendingMachinesService	removeMachine machine: any Promise<any>	Метод видалення вендингу з бази даних
VendingProductComponent	ngOnInit - void	Метод ініціалізації компоненту
VendingProductComponent	ngOnDestroy - void	Метод деструкції компоненту
VendingProductComponent	createProductForm - FormGroup	Метод створення реактивної форми

Продовження таблиці В.2

VendingProductComponent	saveProduct - void	Метод оновлення інформації продукта
VendingProductComponent	addProduct - void	Метод створення нового продукта
VendingProductComponent	onOpenMachine machineProduct: any void	Метод переходу
VendingProductComponent	onAddMachine - void	Метод створення зв’язку між вендингом та продуктом
VendingProductComponent	onMachineRemove event: Event, machineProduct: any void	Метод видалення зв’язку вендинга з продуктом
VendingProductComponent	onMachineProductUpdate event: any, machineProduct: any, input: any void	Метод оновлення кількості продуктів у зв’язку

Продовження таблиці В.2

VendingProductComponent	machineDisplayWith - string	Метод обробка та вивід назви вендингу
VendingProductService	resolve route: ActivatedRouteSnapshot, state: RouterStateSnapshot Promise<any>	Метод попередньої ініціалізації даних для компонента
VendingProductService	getProduct - Promise<any>	Метод отримання продукту
VendingProductService	getMachines - Promise<any>	Метод отримання списку доступних вендивнгів
VendingProductService	saveProduct product: any Promise<any>	Метод оновлення даних про продукт
VendingProductService	addProduct product: any Promise<any>	Метод створення нового екземпляра продукта

Продовження таблиці В.2

VendingProductService	addMachineProduct machine, product Promise<any>	Метод створення зв’язку між продуктом та вендингом
VendingProductService	saveMachineProduct machine, productMachine Promise<any>	Метод оновлення кількості продуктів у зв’язку
VendingProductService	removeMachineProduct productMachine Promise<any>	Метод видалення зв’язку вендингу з продуктом
VendingProductsComponent	ngOnInit - void	Метод ініціалізації компонента
VendingProductsComponent	ngOnDestroy - void	Метод деструкції компонента
VendingProductsComponent	onRemove event: any, product: Product void	Метод видалення продукта

Продовження таблиці В.2

VendingProductsService	resolve route: ActivatedRouteSnapshot, state: RouterStateSnapshot Promise<any>	Метод попередньої ініціалізації даних для компонента
VendingProductsService	getProducts - Promise<any>	Метод отримання списку активних продуктів
VendingProductsService	removeProduct product: any Promise<any>	Метод видалення продукту
VendingStoreComponent	ngOnInit - void	Метод ініціалізації компонента
VendingStoreComponent	ngOnDestroy - void	Метод деструкції компонента
VendingStoreComponent	createStoreForm - FormGroup	Метод створення реактивної форми

Продовження таблиці В.2

VendingStoreComponent	onSaveStore - void	Метод оновлення даних магазина або точки
VendingStoreComponent	onAddStore - void	Метод створення нового екземпляру магазина або точки
VendingStoreComponent	onOpenMachine machine: any Promise<any>	Метод переходу до обраного вендингу
VendingStoreComponent	onAddMachine - Promise<any>	Метод створення зв’язку між магазином та вендингом
VendingStoreComponent	onMachineRemove event: Event, machine: any Promise<any>	Метод видалення зв’язку між магазином та вендингом
VendingStoreComponent	onAddEmployer - void	Метод створення зв’язку між працівником та магазином

Продовження таблиці В.2

VendingStoreComponent	onEmployerRemove event: Event, employer: any void	Метод видалення зв’язку між працівником та магазином
VendingStoreComponent	machineDisplayWith value: any string	Метод обробки та відображення назви магазину
VendingStoreComponent	employerDisplayWith value: any string	Метод обробки та відображення працівника
VendingStoreService	resolve route: ActivatedRouteSnapshot, state: RouterStateSnapshot Promise<any>	Метод попередньої ініціалізації даних для компонента
VendingStoreService	getStore - Promise<any>	Метод отримання обраного магазину магазину
VendingStoreService	getMachines - Promise<any>	Метод отримання активних вендингових машин

Продовження таблиці В.2

VendingStoreService	listUsers - void	Метод отримання пулу користувачів з AWS Cognito
VendingStoreService	getStoreEmployees - Promise<any>	Метод отримання зв’язаних працівників з магазином
VendingStoreService	saveStore store: any Promise<any>	Метод оновлення інформації магазина
VendingStoreService	addStore store: any Promise<any>	Метод створення нового екземпляра магазину
VendingStoreService	addMachine store, machine Promise<any>	Метод створення зв’язку між магазином та вендингом
VendingStoreService	removeMachine machine: any Promise<any>	Метод видалення зв’язку між магазином та вендингом

Продовження таблиці В.2

VendingStoreService	addEmployer employer: any Promise<any>	Метод створення зв’язку між працівником та магазином
VendingStoreService	removeEmployer employer: any Promise<any>	Метод видалення зв’язку між працівником та магазином
VendingStoresComponent	ngOnInit - void	Метод ініціалізації компонента
VendingStoresComponent	ngOnDestroy - void	Метод деструкції компонента
VendingStoresComponent	onRemove event: any, machine: any void	Метод видалення екземпляра магазина